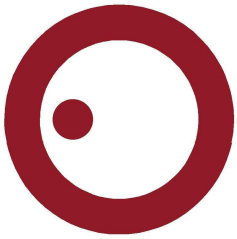


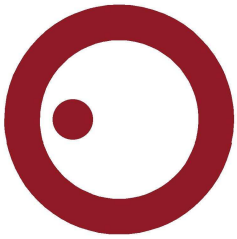
escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

DESENVOLVIMENTO SEGURO DE APLICAÇÃO WEB SMART SIGN

ESTUDANTE FÁBIO DA SILVA BAPTISTA

Leiria, 21 de Março de 2024



escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

DESENVOLVIMENTO SEGURO DE APLICAÇÃO WEB SMART SIGN

ESTUDANTE FÁBIO DA SILVA BAPTISTA
Número: 2213268

Projeto realizado sob orientação do Professor Doutor Sílvio Priem Mendes (smendes@ipleiria.pt), Professor Doutor Paulo Jorge Gonçalves Loureiro (paulo.loureiro@ipleiria.pt) e Professor Doutor Paulo Manuel Almeida Costa (paulo.costa@ipleiria.pt).

Leiria, 21 de Março de 2024

AGRADECIMENTOS

Gostaria de expressar a minha sincera gratidão a todas as pessoas que contribuíram para a realização desta tese. Este trabalho não teria sido possível sem o apoio e incentivo de muitos.

Agradeço aos meus orientadores pela orientação valiosa e dedicação ao longo deste processo.

Muito obrigado a todos.

RESUMO

Esta tese aborda a complexidade inerente do desenvolvimento de uma plataforma inovadora para gestão de sinais rodoviários e *Road Side Unit* (RSU), enquadrada no tema dos *Intelligent Transport System* (ITS). O objetivo consistiu na criação de uma plataforma funcional e segura, capaz de integrar eficazmente os elementos no ambiente rodoviário. O desenvolvimento da plataforma, teve de ultrapassar desafios relacionados com a estrutura das mensagens *In-Vehicle Information Management* (IVIM), a qual exigiu a criação de uma extensa base de dados.

A metodologia adotada incluiu o desenvolvimento seguro da aplicação, incorporando diretrizes de organizações de segurança. Além disso, foi realizada uma análise de várias metodologias de *penetration testing* para identificar vulnerabilidades e para implementar medidas corretivas. Os resultados indicam não apenas a robustez da aplicação, mas também a sua maturidade em termos de segurança.

Este projeto representa um avanço significativo para o ITS, evidenciando a necessidade imperativa de aplicações seguras e de *penetration testing* para garantir a integridade do ecossistema digital. Além disso, destaca-se a importância crítica da cibersegurança na proteção não apenas da aplicação, mas também dos dados criados e manipulados por ela.

Ao refletir sobre a jornada, percebe-se que o trabalho não apenas solucionou os desafios identificados inicialmente, mas também proporcionou uma aprendizagem valiosa em desenvolvimento seguro e exploração de vulnerabilidades.

Palavras-chave: ITS, Sinais Rodoviários, Desenvolvimento *Web*, Cibersegurança e *Penetration Testing*

ABSTRACT

This thesis addresses the inherent complexity of developing an innovative platform for road sign and RSU management within the context of ITS. The goal was to create a functional and secure platform capable of effectively integrating elements into the road environment. The platform development had to overcome challenges related to the structure of IVIM messages, which required the creation of an extensive database.

The adopted methodology included the secure development of the application, incorporating security guidelines from organizational standards. Additionally, an analysis of various penetration testing methodologies was conducted to identify vulnerabilities and implement corrective measures. The results indicate not only the robustness of the application but also its maturity in terms of security.

This project represents a significant advancement for ITS, highlighting the imperative need for secure applications and penetration testing to ensure the integrity of the digital ecosystem. Furthermore, it underscores the critical importance of cybersecurity in protecting not only the application but also the data created and manipulated by it.

Reflecting on the journey, it becomes evident that the work not only addressed the initially identified challenges but also provided valuable insights into secure development and vulnerability exploration.

Keywords: ITS, Road Signs, Web Development, Cybersecurity and Penetration Testing

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

2FA	*Second Factor Authentication.*
ACLs	*Access Control Lists.*
AES	*Advanced Encryption Standard.*
API	*Application Programming Interface.*
ASF	*Application Security Framework.*
BSIMM	*Building Security In Maturity Model.*
CAM	*Cooperative Awareness Message.*
CDN	*Content Delivery Network.*
CI/CD	*Continuous integration/Continuous delivery.*
CIS	*Center for Internet Security.*
C-ITS	*Cooperative Intelligent Transport Systems.*
CORS	*Cross-Origin Resource Sharing.*
CRLF	*Carriage Return Line Feed.*
CRUD	*Create, Read, Update and Delete.*
CSC	*Critical Security Controls.*
CSP	*Content Security Policy.*
CSRF	*Cross-Site Request Forgery.*
CVE	*Common Vulnerability and Exposures.*
CWE	*Common Weakness Enumeration.*
DAST	*Dynamic Application Security Testing.*
DBMS	*Database Management Dystem.*
DENM	*Decentralized Environmental Notification Message.*
DMS	*Driver Management System.*
DNS	*Domain Name System.*

DoS	*Denial of Service.*
EVCSM	*Electric Vehicle Charging Station Management System.*
GIC	*General IVI Container.*
GLC	*Geographic Location Container.*
HIPPA	*Health Insurance Portability and Accountability Act.*
HTML	*HyperText Markup Language.*
HTTP	*HyperText Transfer Protocol.*
HTTPS	*HyperText Transfer Protocol Secure.*
IAST	*Interactive Application Security Testing.*
ID	*Identity.*
IDOR	*Insecure Direct Object References.*
IMS	*Incident Management System.*
IoV	*Internet of Vehicle.*
IP	*Internet Protocol.*
ISECOM	*Institute for Security and Open Methodologies.*
ISSA	*Information Systems Security Association.*
ISSAF	*Information System Security Assessment Framework.*
ITS	*Intelligent Transport System.*
ITS-SCUs	*ITS Station Communication Units.*
ITS-SUs	*ITS Station Units.*
IVI	*In-Vehicle Infotainment.*
IVIM	*In-Vehicle Information Management.*
JSON	*JavaScript Object Notation.*
JWT	*JSON Web Token.*
LDAP	*Lightweight Directory Access Protocol.*

LTE-V	*Long-Term Evolution Vehicular*.
MAP	*Map Message*.
MD5	*Message-Digest*.
MFA	*Multifactor Authentication*.
MITRE	*Massachusetts Institute of Technology Research and Engineering*.
MVC	*Model-View-Controller*.
NIST	*National Institute of Standards and Technology*.
NoSQL	*Not only Structured Query Language*.
NPM	*Node Package Manager*.
NSA	*National Security Agency*.
NVD	*National Vulnerability Database*.
OAuth	*Open Authorization*.
OBU	*On-Board Unit*.
OGNL	*Object-Graph Navigation Language*.
ORM	*Object-Relational Mapping*.
OSINT	*Open-Source Intelligence*.
OSSTMM	*Open Source Security Testing Methodology Manual*.
OWASP	*Open Web Application Security Project*.
PBKDF	*Password-Based Key Derivation Function*.
PCI DSS	*Payment Card Industry Data Security Standard*.
PenTest	*Penetration Testing*.
PHP	*Hypertext Preprocessor*.
PKCS	*Public Key Cryptography Standards 1*.
PMS	*Parking Management System*.
PTES	*Penetration Testing Execution Standard*.
QRCode	*Quick-Response Code*.

RGPD	Regulamento Geral de Proteção de Dados.
RSU	*Road Side Unit*.
SAMM	*Software Assurance Maturity Model*.
SANS	*SysAdmin, Audit, Network and Security*.
SAST	*Static Application Security Testing*.
SEO	*Search Engine Optimization*.
SHA1	*Secure Hash Algorithm 1*.
SIEM	*Security Information and Event Management*.
SOAP	*Simple Object Access Protocol*.
SP	*Special Publication*.
SPA	*Single-Page Application*.
SPAT	*Signal Phase and Timing Message*.
SQL	*Structured Query Language*.
SSL	*Secure Sockets Layer*.
SSRF	*Server Side Request Forgery*.
SSTI	*Server Side Template Injection*.
TC	*Traffic Class*.
TCP	*Transmission Control Protocol*.
TI	Tecnologia da Informação.
TLS	*Transport Layer Security*.
UDP	*User Datagram Protocol*.
URL	*Uniform Resource Locator*.
V2B	*Vehicle-to-Building*.
V2G	*Vehicle-to-Grid*.
V2I	*Vehicle-to-Infrastructure*.
V2P	*Vehicle-to-Peestrian*.
V2R	*Vehicle-to-Roadside*.
V2V	*Vehicle-to-Vehicle*.
V2X	*Vehicle-to-Everything*.

VPN *Virtual Private Network.*

WAF *Web Application Firewall.*

WASC *Web Application Security Consortium.*

WSTG *Web Security Testing Guide.*

XML *Extensible Markup Language.*

XSS *Cross-Site Scripting.*

1 INTRODUÇÃO

No contexto das cidades inteligentes, a segurança rodoviária emerge como uma prioridade vital para garantir não apenas a eficiência, mas, acima de tudo, a segurança dos sistemas de transporte. Este trabalho é uma incursão na criação e implementação de uma plataforma *web*, cujo propósito é gerir sinais rodoviários com tecnologia ITS acoplada. O foco central desta plataforma recai, não apenas na gestão, mas também em planear a implementação estratégica de sinais de trânsito no terreno. Isto não só visa otimizar a segurança nas estradas, mas também permite que os veículos detetem os sinais de forma automática e que mostrem aos condutores, contribuindo significativamente para o avanço tecnológico nas cidades inteligentes.

O primeiro objetivo deste projeto é a criação de uma plataforma *web* robusta que permita a gestão de sinais rodoviários. Esta plataforma não é apenas um sistema de gestão de tráfego, mas uma ferramenta que, por meio da implementação estratégica de sinais, pode aumentar a segurança rodoviária. Alinhando-se com a visão das cidades inteligentes, este projeto insere-se no contexto de ITS, com ênfase na comunicação *Vehicle-to-Infrastructure* (V2I). A aplicação *web* é importante para aprimorar a eficiência do tráfego, proporcionar uma experiência de condução mais segura e contribuir para a condução autónoma.

Para além do desenvolvimento da aplicação, o projeto adotou uma abordagem abrangente relativamente à segurança. Explorando diversas tecnologias relacionadas à segurança, o desenvolvimento seguro da aplicação teve o seu foco na integridade e confidencialidade dos dados. Uma etapa crucial foi a realização de um *Penetration Testing* (PenTest), cujo propósito foi validar não apenas as funcionalidades, mas também a segurança da aplicação contra potenciais ameaças cibernéticas.

1.1 ENQUADRAMENTO

Este projeto abrange diversas áreas de conhecimento, sendo o ITS e a cibersegurança os pilares fundamentais que sustentam todo o desenvolvimento realizado. Ao abordar a segurança rodoviária, a plataforma engloba aspetos como a comunicação entre infraestruturas e veículos, destacando-se como uma contribuição significativa para a eficiência do tráfego nas cidades inteligentes. O desenvolvimento da aplicação baseou-se nas tecnologias Laravel e Vue.js para *backend* e *frontend* respetivamente.

Para a segurança da aplicação foi utilizado como referência o OWASP *top* 10, comparando-o com o *top* 50 das *Common Weakness Enumeration* (CWE). Estas escolhas visam garantir que a aplicação esteja não apenas funcional, mas também resguardada contra potenciais vulnerabilidades. O método de teste de penetração, baseado na *Penetration Testing Execution Standard* (PTES) e na checklist do OWASP, desempenhou um papel crucial na avaliação da segurança da aplicação.

Para compreender integralmente este trabalho, é imperativo adquirir conhecimentos sólidos em ITS. O desenvolvimento da aplicação segue a arquitetura *Model-View-Controller* (MVC), garantindo modularidade e facilitando a sua manutenção.

Também é de salientar que os testes à aplicação foram realizados até a completa identificação e solução de quaisquer problemas eventualmente encontrados na aplicação.

1.2 ESTRUTURA DO DOCUMENTO

Ao longo desta secção é descrita a estrutura do documento, que se divide em seis capítulos fundamentais: Trabalho Relacionado, Análise, Desenvolvimento, *Pentest*, Conclusões e Trabalho Futuro.

No capítulo 2, é realizada uma análise abrangente do estado da arte em ITS, destacando tanto as tendências quanto os desafios presentes no dinâmico panorama das cidades inteligentes. São apresentadas as diversas metodologias de teste de penetração. São incluídas as metodologias *Open Source Security Testing Methodology Manual* (OSSTMM), OWASP *Web Security Testing Guide* (WSTG), *National Institute of Standards and Technology* (NIST) SP 800-115, PTES e *Information System Security Assessment Framework* (ISSAF). Por fim, o capítulo também oferece uma visão consolidada das metodologias de teste de penetração exploradas. Essa análise crítica fornece uma base sólida e abrangente para as etapas subsequentes do desenvolvimento, contribuindo para a abordagem adotada neste projeto.

No capítulo 3, é explorado como o conceito de ITS é integrado e contextualizado no âmbito deste projeto, delineando a sua importância para aprimorar a segurança no tráfego rodoviário. Além disso, o capítulo 3 abrange uma avaliação criteriosa das diversas organizações de segurança que atuam no desenvolvimento seguro de aplicações. No mesmo contexto, é realizado uma comparação de várias metodologias de teste de penetração no âmbito do capítulo 3 para identificar aquela que mais se alinha com as características e requisitos específicos deste projeto.

No capítulo 4, é delineado de forma abrangente todas as etapas e nuances do trabalho realizado, representando a espinha dorsal deste projeto. Inicialmente

são descritas as funcionalidades essenciais da aplicação, especialmente aquelas relacionadas aos ITS. É detalhado como a plataforma foi concebida para gerir sinais rodoviários dinamicamente, visando otimizar não apenas o tráfego, mas também a segurança rodoviária. Paralelamente ao desenvolvimento funcional, o capítulo aborda a implementação de práticas seguras desde as fases iniciais do desenvolvimento, visando mitigar diversas vulnerabilidades potenciais. Cada etapa é descrita, desde a identificação e análise de potenciais riscos até a implementação de contramedidas eficazes.

No capítulo 5, é explorado o processo abrangente de PenTest. Este é um estágio crítico no qual a segurança da aplicação é submetida a rigorosos testes, simulando diversos cenários de ataque. É descrito como foram realizados testes de diversos tipos de ataques, explorando potenciais vulnerabilidades e avaliando a resistência da aplicação a ameaças cibernéticas. Este processo inclui não apenas a identificação de falhas, mas também sugestões de correções e aprimoramentos para garantir a robustez contínua da aplicação em ambientes dinâmicos e adversos.

Por fim, nos capítulos subsequentes, são introduzidas as considerações finais e perspetivas futuras deste trabalho. O capítulo 6, oferecerá uma síntese crítica de todas as descobertas, contribuições e desafios enfrentados ao longo do desenvolvimento deste projeto. Esta secção não apenas recapitulará os objetivos alcançados, mas também proporcionará *insights* valiosos para o entendimento do impacto e relevância do trabalho realizado.

Posteriormente, no capítulo 7, é delineado as possíveis extensões e aprimoramentos que podem ser explorados para evoluir a aplicação. Este capítulo oferecerá uma visão estratégica para futuros desenvolvimentos, incentivando a continuidade do trabalho e a contribuição para a inovação neste domínio dinâmico.

2

TRABALHO RELACIONADO

Neste capítulo, é apresentado e descrito um conjunto de trabalhos representativos sobre o tema deste projeto. Tendo como ênfase sistemas de transporte inteligentes cooperativos, mais conhecidos por *Cooperative Intelligent Transport Systems* (C-ITS), sobre cibersegurança em aplicações *web*, *frameworks* e *standards* de desenvolvimento seguro, bem como metodologias de *penetration testing*.

As especificações C-ITS definem um conjunto de tecnologias avançadas que permitem aos veículos e as infraestruturas comunicarem entre si, visando a melhoria da mobilidade, conforto, segurança e eficiência no meio rodoviário. A tecnologia integra um conjunto de sensores de controlo, análise e tecnologias de comunicação em infraestrutura de viagens [1]. Este conceito teve origem na indústria automóvel e visa aumentar a segurança no trânsito rodoviário, dando origem a várias funções e tarefas no controlo de tráfego, tais como: priorizar serviços de resgates, priorizar os transportes públicos, melhorar as informações disponíveis para os condutores, avisos de perigos, identificação e contagem de veículos, medição de tráfego, mensagens padronizadas e por fim canais de comunicação [2]. As especificações C-ITS e os conceitos associados são abordados na secção 2.1.

Além de explorar avanços em C-ITS para melhorar a mobilidade e segurança rodoviária, este projeto também abrangeu o desenvolvimento de uma aplicação *web* para gestão de sinais rodoviários e RSU.

A segurança informática nos *websites* é extremamente importante com vista à proteção de dados confidenciais, como informação de clientes, informações financeiras e informações comerciais. A segurança informática também visa a proteção contra ataques mal-intencionados, como *malware*, *ransomware* e ataques de *phishing*. Além disso, a segurança informática pode ajudar a evitar violações de dados, como roubo de credenciais de utilizadores ou exposição de dados por falha de segurança informática, que podem ter sérias consequências financeiras e legais. Como tal, é importante que os *websites* sejam implementados com uma estratégia robusta de cibersegurança visando a proteção dos dados e utilizadores [3, 4]. Em resposta foram criadas *frameworks* de cibersegurança essenciais para maximizar a proteção das aplicações de ataques maliciosos, estas serão abordadas na secção 2.2.

Por fim, e para verificar a segurança de um *website*, é necessário realizar um conjunto de testes de segurança com várias ferramentas. Para a realização dos testes

é utilizada uma metodologia de *penetration testing* que visa identificar um conjunto de vulnerabilidades e ameaças que possam ser exploradas por *hackers*. Desta forma será possível avaliar, por meio de um conjunto de métricas, a segurança da aplicação *web*, corrigir as lacunas de segurança e inclusive, realizar comparações entre as ferramentas utilizadas. Inicialmente será realizado um estudo das metodologias de *penetration testing* existentes no mercado para futuramente escolher a metodologia mais adaptada ao projeto. Este estudo será abordado na secção 2.3.

2.1 C-ITS

O C-ITS é um subconjunto de especificações do ITS. O ITS visa melhorar o transporte de rodoviário no que diz respeito à segurança, eficiência, conforto e sustentabilidade. As especificações ITS são geralmente desenvolvidas para atender a um domínio específico de serviço ITS, como transporte público, segurança rodoviária, logística e emergência [5].

Para suportar a interoperabilidade, as especificações C-ITS são desenvolvidas para trocar e compartilhar informações de aplicações ITS de um determinado domínio de aplicação, tais como V2I e *Vehicle-to-Vehicle* (V2V) que serão descritas nesta secção [5].

Os serviços C-ITS baseiam-se na troca de dados entre veículos de qualquer categoria, infraestruturas rodoviárias e urbanas, centros de controlo e serviços na nuvem e até com outros utilizadores da estrada [5].

O C-ITS é um elemento importante no contexto das *smart cities*, pretendem ser cidades eficientes, conectadas e sustentáveis por utilização de tecnologias digitais. Simultaneamente representam uma resposta aos desafios económicos, sociais e políticos que as sociedades pós-industriais enfrentam no início do novo milénio. Através da redução de custos operacionais com auxílio de monitorizações e automações, aumento da conectividade com sensores urbanos e plataformas digitais e priorizar a sustentabilidade ambiental com a gestão de resíduos e transportes públicos. [6].

É necessário combinar múltiplas tecnologias de acesso, protocolos de comunicação com características de desempenho distintas, tais como, alcance de comunicação, largura de banda disponível, atraso de transmissão, qualidade de serviço e segurança [5].

A utilização das várias tecnologias de acesso e protocolos de comunicação requer que as comunicações e os dados permanecerem seguros. Desta forma existiu a necessidade de criar uma arquitetura para o ITS [7]. Esta é abordada na subsecção 2.1.1.

2.1.1 *Arquitetura do ITS*

A conceção da arquitetura ITS visa a gestão da segurança, comunicação e dados associados aos serviços C-ITS [8]. A arquitetura do ITS é especificada no *standard* ISO 21217:2020[9] e apresentada na Figura 1 [5].

A arquitetura ITS é uma área de pesquisa em constante evolução e, por isso, existem diferentes abordagens e modelos propostos por diferentes entidades [7]. No entanto, a ISO 21217[9] é um dos modelos mais reconhecidos e utilizados no contexto de ITS. A ISO 21217:2020[9] propõe uma abordagem em cinco camadas [8].

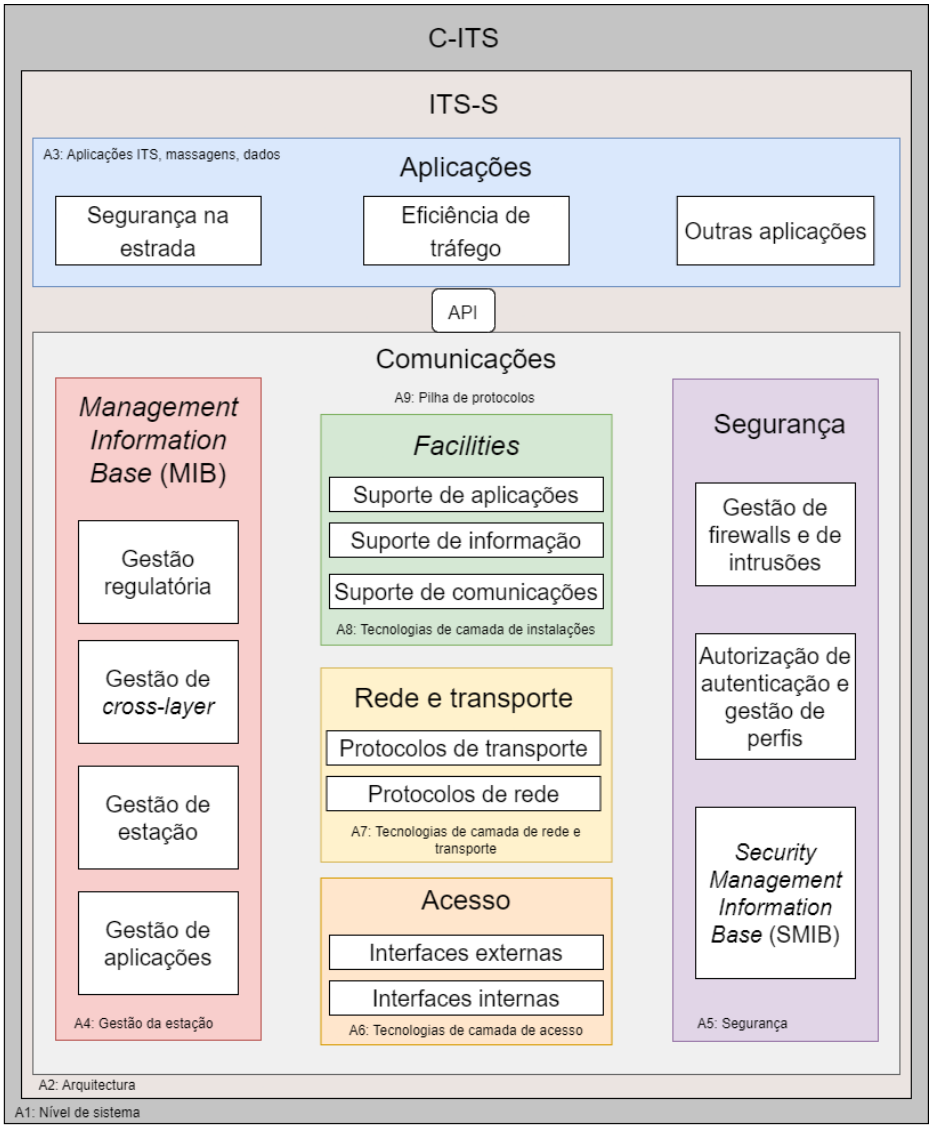


Figura 1: Arquitetura ITS (adaptada de [5])

A arquitetura *station* ITS é dividida em três subcamadas de comunicação independente (a subcamada de acesso à estação ITS, a subcamada de rede e transporte da estação ITS e a subcamada de *facilities* da estação ITS) [10]. A arquitetura

possuí também *cross-layers* adicionais responsáveis pelas atividades de gestão e segurança que suportam as comunicações e as aplicações [11, 12].

A camada de comunicação fornece o meio para transmitir informações entre diferentes entidades do sistema, como a rede de transporte. Esta camada é responsável por especificar os protocolos de comunicação necessários para garantir a interoperabilidade e segurança dos dados transmitidos. Esta camada será descrita na subsecção 2.1.2 [12]. A camada de aplicações é a mais alta na hierarquia de camadas da arquitetura ITS, e é responsável por fornecer serviços específicos aos utilizadores finais, esta camada é abordada na subsecção 2.1.3 [10].

2.1.2 *Camada de comunicação*

A camada de comunicação é uma das camadas mais importantes da arquitetura do ITS, é responsável por garantir a interoperabilidade e a segurança dos dados transmitidas entre os diferentes componentes/entidades do sistema [10, 13].

Esta camada é composta por um conjunto de protocolos de comunicação que permitem que os diferentes componentes do ITS comuniquem entre si [12]. Estes protocolos podem incluir padrões de comunicação sem fio, como *Wi-Fi*, *Bluetooth* e *5G*, bem como padrões de comunicação com fio, como *Ethernet* e *CAN bus* [7, 13].

Adicionalmente, a camada de comunicação também é responsável por garantir a segurança dos dados transmitidos entre os diferentes componentes do sistema [10]. Isso é feito por meio de técnicas de criptografia e autenticação, que garantem que apenas as partes autorizadas possam aceder e modificar os dados transmitidos [13].

Outra função importante da camada de comunicação é gerir o fluxo de dados entre os diferentes componentes do sistema [14]. Isso é feito por meio de técnicas de gestão de tráfego, como a priorização de dados críticos e o controlo de congestionamento [13].

A camada de comunicação do ITS, conforme a ISO 21217 [9], é dividida em três subcamadas: a subcamada de *facilities*, a subcamada de rede e transporte e a subcamada de acesso. Cada uma das subcamadas desempenha um papel fundamental na transmissão de dados entre os diferentes componentes do sistema [9, 15].

A subcamada de *facilities* é responsável pela gestão da infraestrutura física da rede, incluindo cabos, conectores e dispositivos de *hardware* [9, 15, 15].

A subcamada de rede e transporte é responsável por encaminhar os dados entre diferentes redes e dispositivos no sistema de transporte inteligente. Essa subcamada inclui protocolos como o *Internet Protocol* (IP), que permite que os dados sejam enviados entre dispositivos em redes diferentes. E protocolos como *Transmission*

Control Protocol (TCP) e *User Datagram Protocol* (UDP), que garantem a entrega de dados efecientemente [9, 7].

A subcamada de acesso é responsável por gerir a conexão entre os dispositivos do ITS e as redes de transporte. Esta subcamada inclui protocolos de acesso, como *Wi-Fi*, *Bluetooth*, *5G*, e outros padrões de comunicação sem fio, bem como padrões de comunicação com fio, como o *CAN bus* [9, 7, 15].

Estas subcamadas permitem que os diferentes componentes do ITS possam comunicar eficientemente de forma segura, garantindo que o sistema possa funcionar integramente e coordenada para melhorar a mobilidade urbana e a segurança rodoviária [9, 10].

2.1.3 *Camada de aplicações*

A camada de aplicações inclui as aplicações desenvolvidas para fornecer dados e serviços relacionados ao transporte inteligente, como sistemas de gestão de tráfego, sistemas de gestão de estacionamento, sistemas de gestão de transporte público, sistemas de informação ao condutor, entre outros [5, 9].

A camada de aplicações pode ser vista como a *interface* entre os utilizadores finais e as várias tecnologias e sistemas que compõem o ITS [16]. É responsável por fornecer informações aos utilizadores finais, permitindo que os utilizadores tomem decisões informadas sobre a sua viagem, resultando na melhoria da eficiência do transporte [5, 9].

Alguns exemplos de aplicações ITS incluem:

Parking Management System (PMS): permite a gestão eficiente do estacionamento, fornecendo informações sobre a disponibilidade de vagas, permitindo o pagamento eletrónico e o acompanhamento do uso das vagas [17].

Electric Vehicle Charging Station Management System (EVCSM): suporta a gestão de infraestruturas de recarga de veículos elétricos, fornecendo informações sobre a localização das estações de recarga, o status da rede elétrica e a disponibilidade de energia [18].

Incident Management System (IMS): permite a identificação e gestão incidentes na estrada, como colisões, veículos avariados e obstruções, para minimizar interrupções no tráfego [19].

Driver Management System (DMS): fornece informações relevantes aos condutores em tempo real, como alertas de tráfego, condições climáticas, condições de estrada e outras informações importantes [20].

Essas aplicações ITS são desenvolvidas para melhorar a segurança, a eficiência e a sustentabilidade do transporte, fornecendo informações precisas e oportunas aos utilizadores finais, como lotação de estacionamento, acidentes rodoviários ou vias cortadas. Estas decisões auxiliam os utilizadores na tomada de decisões. [5, 14].

2.1.4 *ITS-SU*

As funcionalidades de uma *ITS Station Units* (ITS-SUs) podem ser distribuídas em uma ou mais unidades físicas denominadas por *ITS Station Communication Units* (ITS-SCUs). Isso permite uma implementação flexível mesmo em grandes áreas geográficas, contribuindo para a eficiência do ITS ao facilitar a coordenação e gestão eficaz das operações [5].

Os ITS-SUs podem ser implementados em vários ambientes, incluindo diferentes tipos de veículos, em infraestruturas rodoviárias, na *cloud* ou em dispositivos móveis, conforme ilustrado na Figura 2 [6, 5, 21].

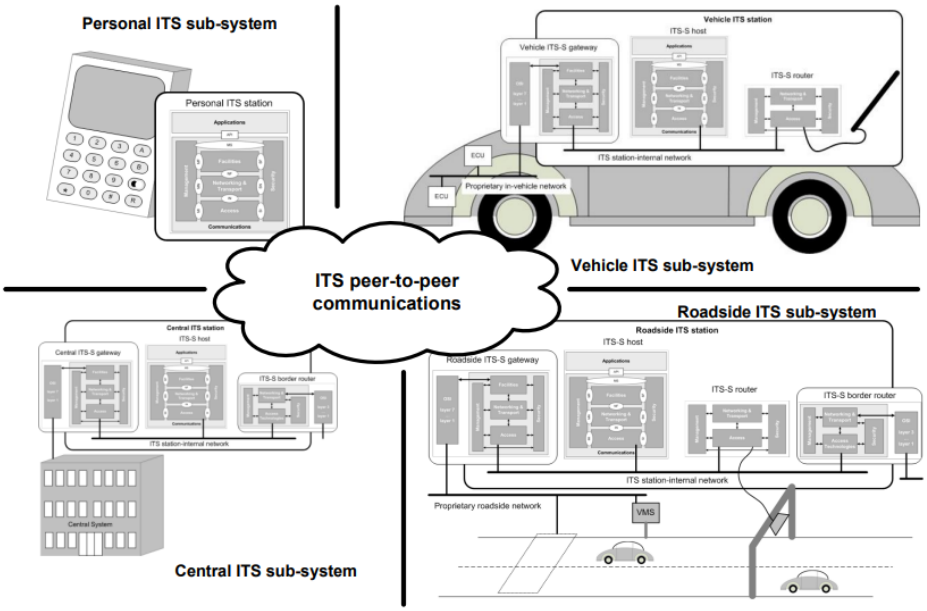


Figura 2: Comunicações *peer-to-peer* entre unidades de estações ITS (fonte [5])

Para assegurar a integridade e a segurança dos serviços providenciados pelo C-ITS e garantir a proteção da vida humana, as ITS-SUs são projetadas para suportar o fornecimento seguro desses serviços, incluído a alocação segura de recursos com acesso priorizado. Os mecanismos de segurança abrangem dois modos operacionais essenciais, a autenticação do remetente de uma mensagem de *broadcast*, utilizadas para disseminação de informações e o estabelecimento e manutenção seguros de sessões [5, 22].

Devido aos requisitos divergentes da multiplicidade de aplicações ITS, diferentes tecnologias de comunicação que são fundamentalmente diferentes podem ser suportadas numa arquitetura ITS-SUs específica. Um princípio de *design* fundamental da arquitetura da *station* ITS é a capacidade de suportar tecnologias de acesso múltiplo e diversos protocolos de comunicação. A arquitetura ITS é, portanto, especificada sem tecnologias de comunicação obrigatórias pré-definidas, pode suportar qualquer tipo de tecnologia existente ou futura, desde cumpra com algumas condições, como requisitos de segurança, eficiência de comunicação ou conformidade com padrões específicos [5, 22].

2.1.5 *RSU*

No contexto de aplicações ITS, o RSU é um dispositivo que faz parte da infraestrutura de comunicação sem fio. A sua principal função é atuar como um ponto de acesso para a rede de comunicação sem fios, permitindo que os veículos e dispositivos conectados possam comunicar com as infraestruturas. O RSU é instalado em locais estratégicos ao longo das estradas, como em postes ou torres, e desempenha um papel fundamental na oferta de conectividade aos dispositivos ITS na sua área de alcance [6].

O RSU pode ser equipado com vários tipos de sensores, como câmaras de vídeo, sensores de deteção de veículos e sensores meteorológicos, que fornecem informações essenciais para os sistemas ITS, como alertas de segurança, controlo de semáforos, dados ambientais ou dados de tráfego em tempo real [6]. A utilização do RSU é fundamental para o desenvolvimento de sistemas ITS seguros e eficientes, permitindo a recolha de informações em tempo real e a comunicação entre veículos e infraestruturas [10]. A perceção do que é um RSU é importante para perceber o objetivo da aplicação *web* desenvolvida.

Para além do RSU existem muitos outros equipamentos no contexto do ITS, como o *On-Board Unit* (OBU). Este é um dispositivo móvel que pode ser instalado nos veículos que permite a comunicação com as infraestruturas do ITS e a transmissão de dados em tempo real [10].

2.1.6 *V2V*

O termo V2V refere-se à comunicação direta entre veículos, sem a necessidade de uma infraestrutura de rede intermediária. Na arquitetura do ITS, a comunicação V2V é considerada uma das tecnologias-chave para melhorar a segurança e eficiência do transporte [23].

A comunicação V2V está incluída na camada de Rede da arquitetura ITS, responsável por permitir a comunicação sem fios entre os dispositivos de comunicação instalados nos veículos [24]. Esses dispositivos geralmente utilizam tecnologias sem fios de curto alcance, como o padrão IEEE 802.11p (também conhecido como Wi-Fi de alta velocidade para veículos) ou o padrão de comunicação móvel de curto alcance *Long-Term Evolution Vehicular* (LTE-V) [23, 25].

A comunicação V2V permite que os veículos troquem informações críticas em tempo real, como dados de posicionamento, velocidade, direção e *status* do veículo. Essas informações podem ser utilizadas para melhorar a segurança do tráfego, como detetar e alertar os condutores sobre potenciais colisões, bem como para melhorar a eficiência do tráfego, para otimizar o fluxo de tráfego em tempo real [23].

A tecnologia V2V desempenha um papel fundamental sendo das principais impulsionadoras do conceito de veículo autónomo, sendo amplamente adotada por fabricantes de veículos e autoridades de transporte em todo o mundo [26].

2.1.7 *V2I*

V2I refere-se à comunicação entre veículos e a infraestrutura de transporte, que inclui semáforos, sinais de trânsito, câmaras, sensores de estacionamento, sistemas de gestão de tráfego e outras tecnologias de transporte. Na arquitetura ITS, a comunicação V2I está incluída na camada de Rede, tal como a comunicação V2V referida anteriormente [23].

A comunicação V2I permite que os veículos comuniquem com a infraestrutura de transporte para obter informações sobre o ambiente de tráfego, incluindo informações sobre o tempo de espera em semáforos, condições de tráfego e alertas de emergência. Além disso, a infraestrutura de transporte pode enviar informações para os veículos, como alertas de congestionamento, rotas alternativas e outras informações relevantes [23].

A comunicação V2I é importante para a implementação de sistemas de transporte inteligentes e pode ajudar a melhorar a segurança, eficiência e sustentabilidade do transporte [26]. Por exemplo, ao receber informações em tempo real sobre as condições de tráfego, os condutores podem tomar decisões mais informadas sobre rotas alternativas, velocidade e outras questões de segurança no trânsito [23].

A comunicação V2I também é para veículos autónomos, por permitir que os veículos comuniquem com a infraestrutura de transporte para obter informações críticas sobre o ambiente de tráfego e tomar decisões em tempo real com base nessas informações [23].

Para além das comunicações V2V e V2I existem outras comunicações, como *Vehicle-to-Everything* (V2X) e *Vehicle-to-Roadside* (V2R) [23]. Na Figura 3 é possível visualizar a utilização das diferentes comunicações.

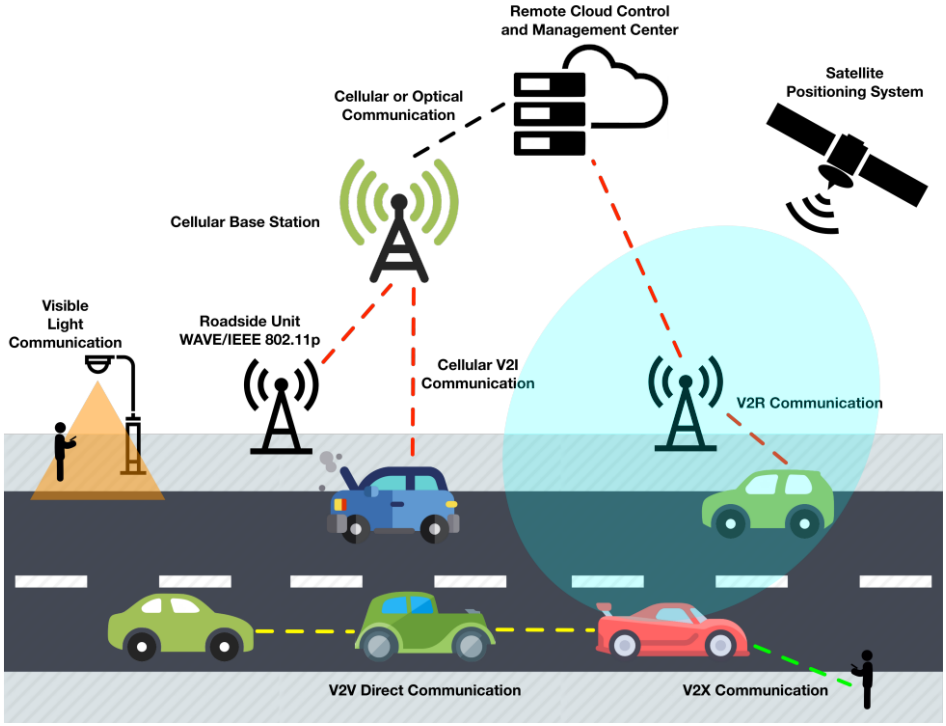


Figura 3: Comunicações no contexto ITS (fonte [25])

2.1.8 *Mensagens*

As mensagens trocadas entre os veículos e a infraestrutura de transporte, como as mensagens *In-Vehicle Infotainment* (IVI), *Cooperative Awareness Message* (CAM) e IVIM, são transmitidas utilizando as comunicações V2V e V2I [13, 25].

A mensagem IVI é uma mensagem de dados que fornece informações sobre o sistema de *infotainment* (informação e entretenimento) de um veículo. Essas mensagens podem incluir detalhes sobre música, navegação, comunicação e outras funcionalidades de entretenimento disponíveis no sistema do veículo [25]. Essas mensagens são geralmente transmitidas por meio de redes sem fio, como Bluetooth ou Wi-Fi, sendo recebidas pelo dispositivo de comunicação do veículo. A mensagem IVI não é uma mensagem crítica para a segurança e é geralmente utilizada para fins de entretenimento e conforto [11].

A mensagem CAM é uma mensagem de dados crítica para a segurança que fornece informações sobre os *status* do veículo, como velocidade, posição, direção e aceleração [11]. Essas mensagens são transmitidas para melhorar a segurança

rodoviária, abrangendo pedestres e veículos, permitindo que os veículos compartilhem informações em tempo real sobre as condições da estrada e outros eventos potencialmente perigosos [13, 25].

A mensagem IVIM proporciona informações essenciais para a condução e segurança do veículo, indo além de simples recursos de entretenimento, ao contrário da mensagem IVI. Isso pode incluir informações sobre o status do veículo, como velocidade, temperatura do motor, nível de combustível, pressão dos pneus, entre outros [11]. O objetivo principal do IVIM é fornecer ao condutor informações relevantes em tempo real para uma condução mais segura e eficiente [25].

A estrutura da mensagem IVIM é muito importante neste projeto e é uma peça fundamental para relacionar os conceitos do ITS com o desenvolvimento da aplicação *web*. A estrutura é composta por campos obrigatórios e opcionais [25]. Na Figura 4 é possível observar um exemplo de estrutura da mensagem IVIM, neste caso com três *containers* (IVI Management, *Geographic Location Container* (GLC) and IVI Application).

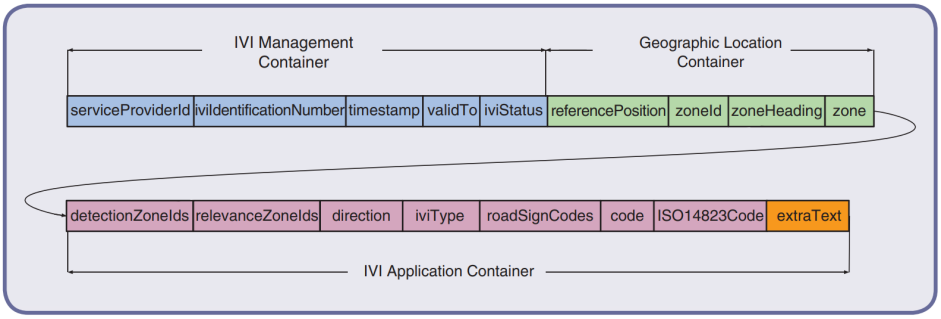


Figura 4: Um exemplo de estrutura IVIM (fonte [25])

À medida que mais veículos e infraestruturas de transporte se tornam equipados com dispositivos de comunicação, espera-se que o número e a variedade de mensagens trocadas aumentem, permitindo uma melhor coordenação do tráfego e uma experiência de condução mais segura e eficiente [11, 25].

Com a crescente adoção de dispositivos de comunicação em veículos e infraestruturas, há uma expectativa de aumento na utilização das mensagens referidas anteriormente. Essa expansão visa contribuir para uma melhoria na experiência de condução, tornando-a mais segura e eficiente para o ambiente rodoviário.

2.2 ORGANIZAÇÕES DE SEGURANÇA

A exploração das inovações e conceitos em ITS e a eficiente troca de mensagens para aprimorar a segurança rodoviária na primeira secção deste capítulo conduz agora à análise da implementação prática dessas tecnologias. Destaca-se a cria-

ção da aplicação *web* dedicada como parte fundamental deste projeto, servindo como plataforma central para o desenvolvimento de sinais rodoviários e RSU. Essa iniciativa exige uma abordagem rigorosa em termos de segurança da informação, garantindo a proteção de dados sensíveis, preservando a integridade das informações e assegurando a disponibilidade contínua dos serviços.

Nesta secção, serão exploradas as melhores práticas e orientações de organizações especializadas em segurança, visando atender aos requisitos de segurança e conformidade relacionados aos dados do ITS criados e mantidos na aplicação *web*.

Inicialmente foram realizadas várias pesquisas para perceber qual a melhor forma de fortalecer a segurança das aplicações *web*. De acordo com os resultados obtidos, verificou-se a existência de várias organizações que desenvolvem *frameworks* e *standards* de segurança para a implementação e desenvolvimento seguro de aplicações *web*. Ao longo das próximas secções serão analisadas várias normas encontradas para desenvolver uma aplicação *web* segura.

É importante perceber que as alternativas das *frameworks* e *standards* que vão ser abordadas podem ter diferentes áreas de foco e podem não cobrir os mesmos tópicos. Por isso, é importante avaliá-las conforme as necessidades específicas do projeto envolvido. Na seguinte lista de tópicos são descritas as várias alternativas que irão ser abordadas.

- OWASP *top* 10: É uma lista dos 10 riscos de segurança de aplicações *web* mais críticos determinados pela comunidade OWASP. A versão mais recente é a de 2021 [27].
- *SysAdmin, Audit, Network and Security* (SANS): O instituto SANS publica e atualiza documentação cibernética, onde é possível analisar os *controls* do *Center for Internet Security* (CIS) ou conferir os CWE mais importantes para desenvolver uma aplicação segura [28].
- CIS *controls*: O CIS publica um conjunto de controlos de segurança críticos que as organizações podem utilizar para se protegerem de ameaças digitaís [29].
- CWE: A comunidade do CWE desenvolve uma lista de vulnerabilidades de *software* e *hardware* que serve como uma linguagem comum, uma medida para ferramentas de segurança e como uma linha base para esforços de identificação, mitigação e prevenção de fraquezas [30].

A seguir, irá ser abordada cada uma das *frameworks* e *standards* individualmente para perceber em que consistem e quais são os seus propósitos.

2.2.1 *OWASP*

O OWASP é um conjunto de *guidelines open source* que fornecem medidas preventivas e orientação para desenvolvimento de aplicações *web*, administradores de sistema e profissionais de segurança [31].

O OWASP fornece uma ampla variedade de recursos de segurança, incluindo ferramentas, materiais educacionais e práticas recomendadas. O OWASP criou uma lista de *top* 10 de riscos de segurança mais críticos associados a aplicações *web*. Esta lista é constantemente atualizada e fornece uma visão detalhada dos problemas de segurança mais comuns e as ações necessárias para mitigá-los [31, 32].

Além disso, o OWASP fornece uma variedade de controlos de segurança que devem ser implementados para proteger os *websites*. Esses controlos incluem validação de entrada, autenticação e autorização, gestão de sessões, criptografia, registo, auditoria e muito mais. O OWASP ainda especifica uma variedade de padrões e diretrizes para garantir a segurança das aplicações *web* [32, 33].

A versão mais recente do OWASP *top* 10, lançada em 2021, apresenta notáveis atualizações em comparação com a edição de 2017. Foram introduzidas 3 novas categorias, houve alterações de designação e âmbito em quatro categorias existentes, além de consolidações significativas no próprio *top* 10 de 2021. Estas modificações estão detalhadas na Figura 5 [34].

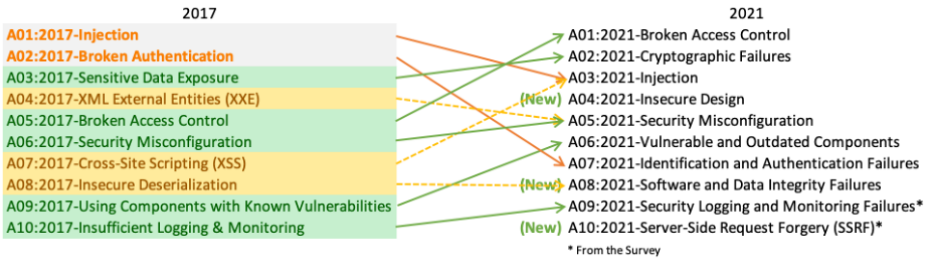


Figura 5: Alterações do OWASP *top* 10 2017 para 2021 (fonte [34])

Foi realizado um conjunto específico de pesquisas por trabalhos relacionados que utilizem o OWASP para desenvolvimento seguro e de certa forma para entender o seu funcionamento, bem como, ferramentas que podem ser utilizadas mutuamente e como implementar testes e análises de vulnerabilidades. Como resultado, foram encontrados diversos documentos relacionados com o OWASP que a seguir serão abordados sucintamente.

O artigo [32] explica o funcionamento do OWASP, ajuda a compreender as vulnerabilidades comuns em aplicações *web* e como proteger os dados contra ciberataques. Não é utilizado num caso prático, porém explica cada vulnerabilidade individualmente com exemplos teóricos.

A página *web* [33] da Synopsys explica para que serve o OWASP e a lista do *top* 10. Abrange cada tópico da lista apresentando uma descrição, um exemplo e uma solução, todos eles com um vídeo educativo.

O documento [35] é um trabalho similar que aborda igualmente o OWASP, mas neste caso o trabalho desenvolvido foi realizado um cenário real com cinco setores do governo de Bangladesh. Adicionalmente ao OWASP, utilizaram ferramentas como BurpSuite [36], ZAP [37] e Netsparker (atualmente denominado por Invicti)[38] para identificar vulnerabilidades recorrentes em aplicações *web*. A análise comparativa dos dados criados por estas ferramentas revelou perceções valiosas sobre a eficácia de cada ferramenta na deteção e categorização das vulnerabilidades. Os resultados das comparações oferecem uma compreensão mais profunda das capacidades das ferramentas na identificação e resolução de falhas específicas do OWASP *top* 10.

Noutra perspetiva de utilização do OWASP, o artigo [39] destaca a importância de realizar testes de segurança nas aplicações *web* para avaliar o nível de vulnerabilidades dos recursos disponibilizados. Esta ênfase destaca a necessidade crucial de incorporar avaliações de segurança, uma prática muitas vezes negligenciada. Consequentemente, é referido a execução de um teste de penetração de *grey box* utilizando o método OWASP *top* 10 de 2021 e a ferramenta OWASP ZAP. O objetivo deste teste é dividido em várias etapas, tais como, recolher informações da aplicação *web*, realizar verificações automáticas, explorar os resultados obtidos, relatar e por último fornecer recomendações. Neste caso prático de utilização do OWASP foram encontradas vulnerabilidades relacionadas com A01 — *Broken Access Control*, A03 — *Injection*, A05 — *Security Misconfiguration* e A08 — *Software and Data Integrity Failures*.

A partir da documentação oficial do CWE [40] é possível consultar a associação do *top* 10 do OWASP com as várias CWE, isto permite rapidamente descobrir quais são as CWE associadas a cada vulnerabilidade critica do OWASP, como está representado na Figura 6 [41, 40].

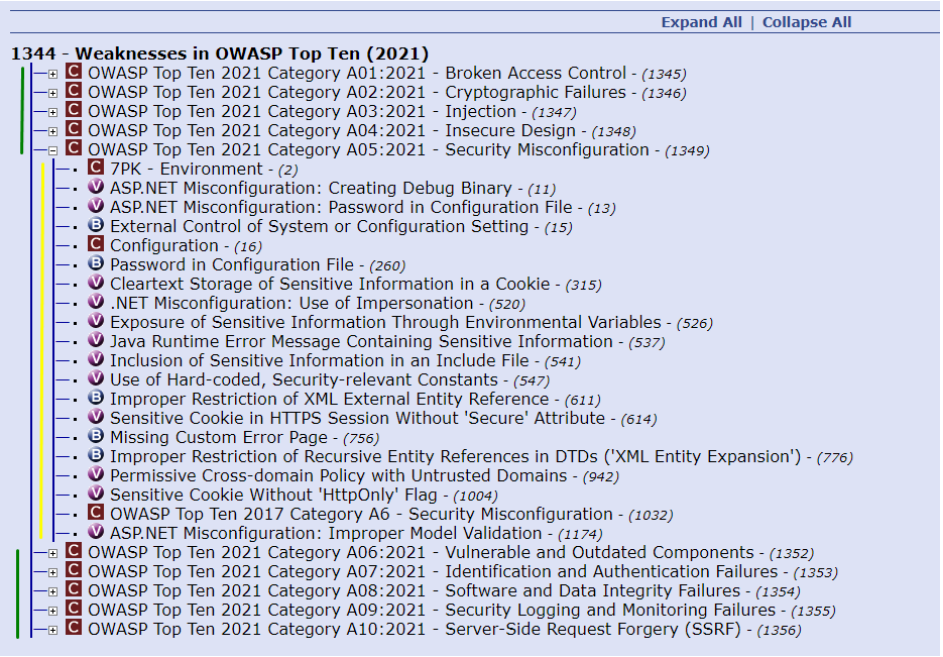


Figura 6: CWE no *top* 10 do OWASP (fonte [40])

Desta forma é possível observar a descrição, relação, plataformas aplicáveis e até exemplos demostrativos de cada CWE. Na Figura 7 está um exemplo detalhado de uma CWE.

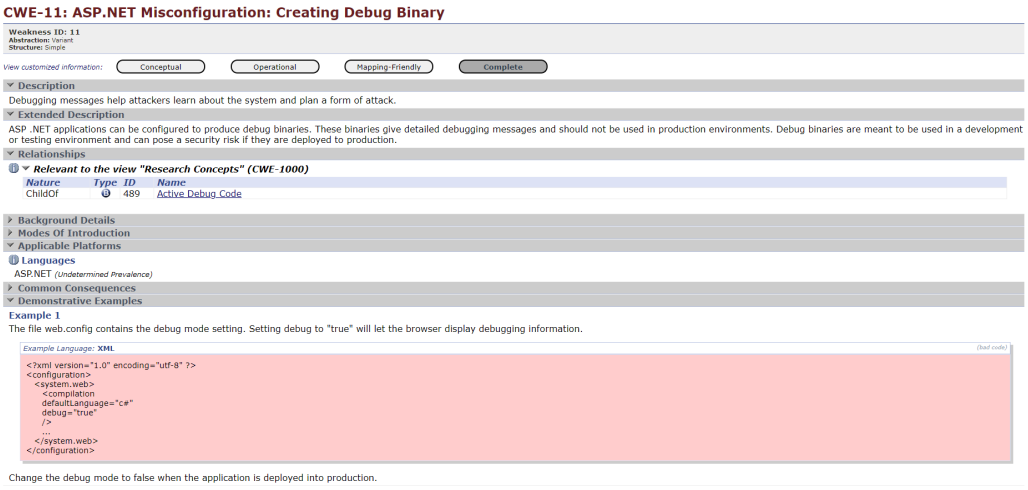


Figura 7: Exemplo de uma CWE (fonte [42])

Com esta informação é possível desenvolver uma aplicação mais segura e ter a perceção de como as várias vulnerabilidades podem ser exploradas pelos atacantes.

2.2.2 *SANS*

O Instituto SANS é uma organização de pesquisa e educação. Atualmente disponibiliza por volta de vinte diferentes categorias de cursos na área de cibersegurança, tais como: *Cyber Defense*, *Open-Source Intelligence* (OSINT), *Penetration Testing and Red Teaming*, *Purple Team*, entre outros cursos de cibersegurança. Também oferece eventos grátis sobre cibersegurança, *webcasts*, *podcasts*, artigos científicos, *blogs* e *newsletters*, na área de cibersegurança. Devido à existência de vários programas de cibersegurança de grande importância, conseguem influenciar mais de 165.000 profissionais de cibersegurança em todo o mundo [28].

Os SANS CIS *controls* ou os controlos do centro de segurança na *internet*, são um conjunto recomendado de ações para a segurança de informação que fornecem normas específicas e acionáveis de impedir os ataques mais difundidos e perigosos da atualidade. Os controlos são aplicáveis a todos os tipos de empresas e organizações de Tecnologia da Informação (TI) e analisam todos os dados sobre ataques. Atualmente o SANS utiliza o CIS *controls*, anteriormente conhecidos por CIS *Critical Security Controls* (CSC), como política de recomendações para proteção da segurança da informação [43].

De momento o CIS utiliza a versão 8 dos controlos, esta atualização foi realizada a 18 de maio de 2021. O SANS oferece suporte ao CIS *controls* com treino, pesquisa e certificações. O CIS disponibiliza toda a informação necessária para a perceção do funcionamento e das alterações entre as versões dos controlos [43].

O SANS CIS 18 é um ponto de partida flexível, aplicável a praticamente qualquer organização, independentemente do tamanho, setor, geografia ou comercial. Desenvolvidos e mantidos por um grupo internacional de organizações, agências governamentais e especialistas em segurança, os controlos são priorizados para proteger a infraestrutura e os dados da organização, fortalecendo o sistema de defesa da organização por meio de proteção e monitorização automatizada contínua. Os 18 principais controlos são mapeados para os controlos do NIST, bem como para as prioridades do *National Security Agency* (NSA). O objetivo dos 18 principais controlos do SANS CIS é proteger os ativos, infraestrutura e informações essenciais. Os controlos também auxiliam a identificar as vulnerabilidades na rede, fortalecer a postura defensiva na organização e monitorizar as informações confidenciais [43].

A análise do CIS *controls* será realizada na subsecção 2.2.3, onde será analisado o estudo do CIS, os seus controlos e as versões associadas aos mesmos. Relembrando que o SANS utiliza os controlos do CIS, apenas adiciona a oferta de treino, pesquisas e certificações para a utilização dos controlos do CIS.

O SANS oferece uma variedade de cursos de treino em segurança da informação, muitos dos quais são baseados na lista CWE e nas melhores práticas de segurança. Esses cursos ajudam a preparar os profissionais de segurança para identificar, prevenir e mitigar vulnerabilidades de segurança em *software* [44]. O CWE é abordado na subsecção 2.2.4.

2.2.3 *CIS controls*

O CIS é uma organização sem fins lucrativos que fornece recursos e soluções para melhorar a segurança informática. O CIS oferece ferramentas, melhores práticas, padrões e serviços de segurança para ajudar as organizações a proteger os sistemas e dados contra ameaças cibernéticas. [45, 46]

Atualmente, o CIS está na versão 8, com uma lista com 18 recomendações de segurança. Esses controlos cobrem uma ampla gama de tópicos de segurança, incluindo privacidade de dados, gestão de ativos, deteção e resposta a incidentes e muito mais [45, 46].

Além da avaliação de controlo de segurança, o CIS também oferece soluções de segurança, como a ferramentas de gestão de segurança gratuitas, como o CIS-CAT[47], que auxilia as organizações a implementar os controlos de segurança recomendados pelo CIS. O CIS também oferece treino e certificações de segurança informática, bem como colaboração com organizações e agências governamentais para reforçar a segurança em todo o mundo [46, 48].

A versão 8 do CIS foi atualizada para acompanhar as mudanças e as novas ameaças no contexto da segurança informática. A equipa do CIS trabalha constantemente para fornecer recursos e soluções de segurança atualizadas e eficazes para ajudar as organizações a proteger os seus sistemas e dados contra ameaças [48]. Na Figura 8 estão representadas as alterações entre versões. A versão mais recente foi lançada na *RSA Conference*[49] em 18 de maio de 2021 [45].

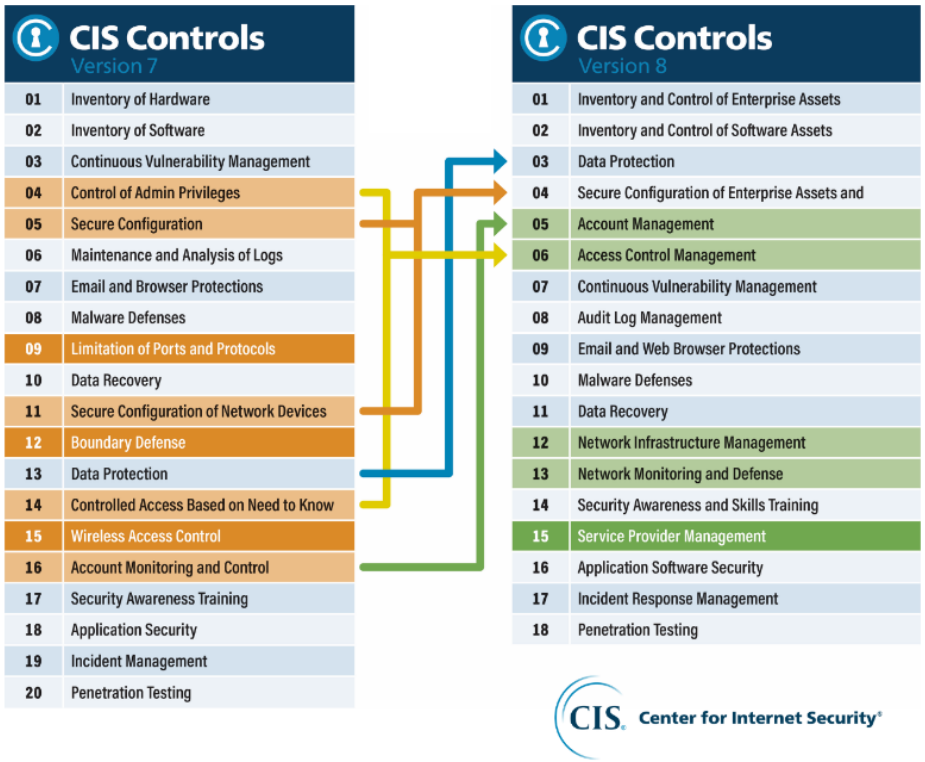


Figura 8: Versão 7 e 8 do CIS *controls* (fonte [45])

2.2.4 *CWE*

O CWE é um catálogo de vulnerabilidades comuns de segurança de *software*, mantido pelo MITRE Corporation. O objetivo do CWE é fornecer um vocabulário comum para descrever e categorizar vulnerabilidades de segurança em *software*, permitindo que programadores, auditores e pesquisadores discutam e trabalhem juntos em soluções de segurança. O CWE é uma lista padronizada de tipos de vulnerabilidades de segurança de *software*, cada uma com uma descrição detalhada, exemplos de código vulnerável e possíveis consequências. O uso do CWE pode ajudar a melhorar a qualidade da segurança de *software* e orientar a priorização de correções de segurança [50].

O CWE é amplamente utilizado pela comunidade de segurança da informação para identificar e mitigar vulnerabilidades de segurança em *software*. É utilizado por organizações, desde pequenas *startups* até grandes empresas de tecnologia, para identificar e corrigir vulnerabilidades de segurança nos seus sistemas. Além disso, o CWE é utilizado em muitos tipos de *software*, incluindo sistemas operativos, aplicações *desktop*, aplicações móveis e *websites* [50, 51, 52].

A utilização do CWE em *websites* é especialmente importante, já que as vulnerabilidades em *websites* podem levar a ataques de segurança informática e violações

de dados. O CWE pode ser utilizado para identificar vulnerabilidades de segurança em *websites*, como vulnerabilidades de injeção SQL, *Cross-Site Scripting* (XSS) e autenticação e autorização inadequadas. Ao identificar e corrigir essas vulnerabilidades, os programadores podem proteger os *websites* contra ataques e proteger os dados dos utilizadores [51, 52].

O OWASP utiliza o CWE como uma das principais fontes de informação para desenvolver a lista *top* 10. Muitas das vulnerabilidades no *top* 10 do OWASP estão diretamente relacionadas às vulnerabilidades de segurança informática identificadas pelo CWE, como injeção SQL, XSS e autenticação e autorização inadequadas. Ao utilizar o CWE em conjunto com o *top* 10 do OWASP, os programadores podem obter uma visão mais abrangente das principais vulnerabilidades de segurança em aplicações *web* e desenvolver soluções mais seguras [53].

2.3 METODOLOGIAS DE *PENETRATION TESTING*

Adicionalmente a estas *frameworks* e *standards* de segurança que ajudam a desenvolver uma aplicação *web* segura, é necessário testar e avaliar a segurança da aplicação *web* construída. Para isso existem algumas metodologias e *standards* de PenTest que irão ser sucintamente analisadas. De acordo com a informação pesquisada foi percebido que existem essencialmente cinco metodologias ou *standards* principais, sendo os seguintes [54, 55]:

- OSSTMM: O OSSTMM, é um guia aberto para testes de segurança que define uma metodologia sistemática e rigorosa para avaliar a segurança de um sistema. É composto por uma série de procedimentos que visam identificar possíveis vulnerabilidades em sistemas, redes e aplicações. O OSSTMM inclui uma variedade de técnicas de teste, que vão desde análises de vulnerabilidades automatizadas a testes manuais detalhados, e visa fornecer uma visão completa da postura de segurança de um sistema. [56].
- OWASP WSTG: O OWASP inclui muitos projetos, como anteriormente descrito o *top* 10 do OWASP que serve para proteger aplicações *web*. Será analisado um outro projeto do OWASP, o OWASP WSTG que é um guia de testes de segurança na *web* que fornece diretrizes abrangentes para testes de segurança em aplicações *web* e destina-se a auxiliar os programadores a planear e executar testes de segurança em aplicações *web*, bem como ajudar os programadores a implementar as melhores práticas de segurança ao projetar e desenvolver aplicações *web* seguras [57].
- NIST SP 800-115: É um padrão do Instituto Nacional de Padrões e Tecnologia dos Estados Unidos para testes de penetração. Fornece uma estrutura para

conduzir testes de penetração em sistemas de informação, incluindo quatro etapas: planeamento, descoberta, ataque e análise de resultados [58].

- PTES: É uma metodologia amplamente utilizada que fornece uma estrutura abrangente para testes de penetração. Inclui sete etapas, as quais são: planeamento, inteligência e recolha de informação, modelagem de ameaças, análise de vulnerabilidades, exploração, pós-exploração e relatório [59].
- ISSAF: É uma estrutura de testes de penetração que fornece diretrizes para realizar testes de segurança em sistemas de informação. O ISSAF cobre todas as fases do teste de penetração, desde a fase de planeamento até à entrega do relatório [60].

2.3.1 *OSSTMM*

O OSSTMM é um guia aberto e gratuito que estabelece padrões para testes de segurança. Foi desenvolvido para fornecer um método abrangente, flexível e prático para implementar testes de segurança em sistemas de informação. O manual é mantido pelo *Institute for Security and Open Methodologies* (ISECOM), uma organização sem fins lucrativos que promove o uso de metodologias de segurança abertas e de código aberto [54].

O OSSTMM é baseado num modelo de ameaças que se concentra na avaliação dos controlos de segurança existentes num sistema de informação. A *framework* define controlos de segurança em seis áreas de teste: informações, pessoas, processos, tecnologia, ambiente físico e comunicações. O OSSTMM oferece uma abordagem sistemática e estruturada para avaliar a eficácia dos controlos de segurança em cada uma dessas áreas [61].

O manual oficial *Open Source Security Testing Methodology Manual (OSSTMM) Version 3* é dividido em várias secções, cada uma focando num aspeto diferente do teste de segurança. Essas secções incluem planeamento e preparação para o teste, recolha de informações sobre o alvo do teste, identificação de vulnerabilidades, avaliação de riscos, exploração de vulnerabilidades, confirmação de vulnerabilidades, cobertura de testes e relatório de resultados [61].

Além disso, o OSSTMM também fornece orientação sobre a realização de testes de segurança de rede, testes de segurança de aplicações, testes de segurança de redes e testes de segurança física. O manual inclui ferramentas e técnicas específicas para cada uma dessas áreas de teste [61].

Uma das principais vantagens do OSSTMM é a sua flexibilidade. Pode ser adaptado para atender às necessidades de diferentes organizações e cenários de teste. Além disso, o manual é atualizado regularmente para refletir as mudanças

na paisagem de segurança da informação e para incorporar novas ferramentas e técnicas de teste [61].

Em resumo, o OSSTMM é uma metodologia de teste de segurança aberta e flexível que fornece um modelo de ameaças baseado em seis áreas de teste: informações, pessoas, processos, tecnologia, ambiente físico e comunicações. Oferece ainda uma abordagem estruturada para avaliação da eficácia dos controlos de segurança em cada uma destas áreas e fornece orientação sobre como realizar diferentes tipos de testes de segurança [61].

2.3.2 *OWASP WSTG*

O OWASP WSTG é um guia abrangente e prático para testes de segurança de aplicações *web*. O guia foi desenvolvido pela comunidade OWASP e destina-se a ajudar programadores de segurança informática a planear e executar testes de segurança em aplicações *web*, bem como ajudar os programadores a entender as melhores práticas de segurança ao projetar e desenvolver aplicações *web* seguras [54].

O guia *OWASP Web Security Testing Guide (v4.2)* é composto por 13 capítulos que cobrem várias áreas de segurança da web, incluindo autenticação, autorização, gestão de sessões, gestão de erros, validação de entrada, manipulação de criptografia, segurança do servidor e gestão de configuração. Cada capítulo contém um conjunto de atividades de teste, bem como informações de fundo e recomendações para mitigar as vulnerabilidades descobertas [62].

O manual é uma ferramenta útil para os programadores de segurança informática, ao ajudar a garantir que todos os aspetos relevantes de segurança da *web* sejam abordados durante os testes. Além disso, fornece um conjunto de atividades de teste práticas que podem ser utilizadas como um ponto de partida para testes de segurança [62].

O guia também é útil para programadores de aplicações da *web*, ao fornecer informações sobre as melhores práticas de segurança que podem ser integradas ao processo de desenvolvimento. Isso pode ajudar a garantir que a aplicação seja desenvolvida com a segurança em mente desde o início e ajude a mitigar os riscos de vulnerabilidades de segurança [62].

Além disso, o guia é projetado para ser compatível com várias *frameworks* de testes de segurança de aplicações *web* e pode ser utilizado em conjunto com outras ferramentas de teste de segurança para fornecer uma cobertura mais completa de segurança de aplicações *web* [62].

Em suma, o OWASP WSTG é uma ferramenta valiosa para programadores de segurança informática e de aplicações *web* que desejam garantir que as suas aplicações sejam seguros e protegidos contra possíveis vulnerabilidades. Com o seu conjunto abrangente de atividades de teste e recomendações de melhores práticas, o WSTG pode ajudar a garantir que a segurança seja uma consideração crítica em todas as fases do ciclo de vida de desenvolvimento das aplicações [62].

O OWASP fornece a seguinte página *web*, *OWASP Testing Guide v4.2 - Penetration Testing Methodologies* [63], onde são descritas as mesmas metodologias de PenTest de forma resumida. Sendo esta uma boa prática para comparar as diferentes metodologias na ajuda da escolha da mais adequada.

2.3.3 *NIST SP 800-115*

O NIST *Special Publication* (SP) 800-115 é um guia técnico para testes e avaliações de segurança da informação. Fornece diretrizes e práticas recomendadas para avaliar a segurança de sistemas, redes, aplicações e outros ativos de informação [54].

O documento *Technical guide to information security testing and assessment*, inicia a descrição da importância dos testes de segurança e os objetivos a serem alcançados por meio desses testes. Também aborda as quatro etapas envolvidas no processo de teste e avaliação: planeamento, descoberta, ataque e análise de resultados e relatório. A Figura 9 representa as quatro fases do *penetration testing* [64].

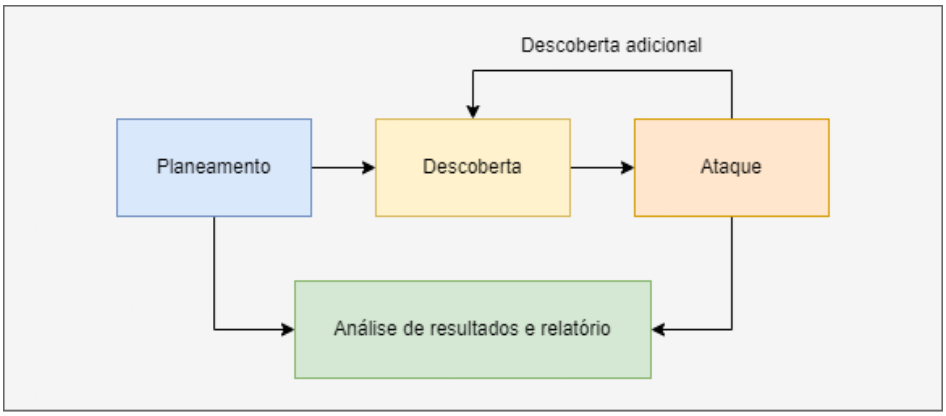


Figura 9: Fases do *penetration testing* do NIST (adaptado de [64])

Em seguida, o mesmo documento descreve as técnicas de teste comuns, incluindo testes de penetração, análise de vulnerabilidades, testes de conformidade e testes de continuidade de negócios e recuperação de desastres. O guia também fornece informações sobre como selecionar as ferramentas e técnicas apropriadas para atender às necessidades específicas de teste de segurança. [64]

Além disso, o NIST SP 800-115 aborda a importância da documentação adequada durante o processo de teste e avaliação. Descreve ainda os tipos de documentação que devem ser mantidos, incluindo o plano de teste, resultados do teste e relatório final. O documento também discute considerações éticas e legais que devem ser consideradas durante o processo de teste [64].

Em resumo, o NIST SP 800-115 é uma fonte valiosa de informações para profissionais de segurança da informação que tencionam implementar práticas recomendadas de teste e avaliação de segurança. Fornece orientações detalhadas sobre o processo de teste e as técnicas e ferramentas comuns utilizadas na avaliação da segurança da informação [64].

2.3.4 *PTES*

O PTES é um padrão de execução de testes de penetração, desenvolvido para fornecer uma estrutura consistente e orientada para profissionais de segurança cibernética que realizam testes de penetração em sistemas e redes [65].

O objetivo do PTES é fornecer um padrão comum para a realização de testes de penetração, de forma que os resultados sejam consistentes e confiáveis, independentemente da equipa de testes ou organização que realiza o teste. Isso permite que as organizações tenham uma melhor compreensão dos riscos de segurança nas suas redes e sistemas, e possam tomar medidas para mitigar esses riscos [65].

O documento intitulado por *Penetration Testing Execution Standard (PTES)* é um guia detalhado para a realização de testes de penetração. Este guia é uma referência para a realização de testes de penetração em sistemas e redes, e fornece uma estrutura para a execução desses testes [59].

O guia começa com uma introdução aos testes de penetração, explicando o conceito e a sua importância. Discute ainda os diferentes tipos de testes de penetração que podem ser realizados, incluindo testes de rede, testes de aplicações e testes de engenharia social [59].

O guia também discute as sete etapas envolvidas na realização de um teste de penetração. Isso inclui o pré-compromisso, recolha de informação, modelação de ameaças, análise de vulnerabilidades, exploração de vulnerabilidades, pós-exploração e documentação de resultados [59]. A Figura 10 representa as etapas do PenTest.

Figura 10: Fases do *penetration testing* do PTES (adaptado de [65])

O manual fornece um conjunto de diretrizes e melhores práticas para a realização de cada uma dessas etapas, incluindo como identificar alvos, como avaliar a segurança física e lógica, como escolher ferramentas de teste de penetração e como conduzir testes de engenharia social [59].

O guia também discute os diferentes tipos de relatórios que podem ser criados a partir de um teste de penetração e como esses relatórios podem ser utilizados para melhorar a segurança do sistema ou da rede testada [59].

Por fim, o documento discute as implicações éticas e legais da realização de testes de penetração, incluindo a necessidade de obter permissão por escrito antes de realizar um teste e as possíveis consequências legais caso um teste seja realizado sem autorização [59].

O *Penetration Testing Execution Standard (PTES)* é um guia abrangente para de testes de penetração, oferecendo informações valiosas para profissionais de cibersegurança e outros profissionais envolvidos na proteção de sistemas e redes contra ataques maliciosos [59].

2.3.5 *ISSAF*

A ISSAF é uma *framework* para avaliação de segurança da informação que fornece um conjunto de diretrizes e procedimentos para auxiliar na identificação, análise e gestão de riscos de segurança em sistemas de informação.

Desenvolvido pela *Information Systems Security Association* (ISSA), a ISSAF é uma abordagem completa que ajuda a padronizar a avaliação de segurança de sistemas, fornecendo uma estrutura para as organizações poderem implementar avaliações consistentes e precisas.

O objetivo principal da ISSAF é auxiliar as organizações a entenderem os seus riscos de segurança, identificando vulnerabilidades e ameaças nos seus sistemas de informação, e, a partir disso, fornecer recomendações de ações para melhorar a segurança dos sistemas.

O documento é extenso com mais de 1.200 páginas. Esta *framework* é ideal para organizações individuais, permitindo a criação de planos de teste personalizados. É importante perceber que o ISSAF vai além do PenTest, também abrange o desenvolvimento de ferramentas que podem ser usadas para educar outras pessoas que têm acesso a uma rede [66].

Apesar desta *framework* ser bastante robusta e seja uma fonte de referência valiosa, deixou de ser atualizada. Portanto, é provável que se torne cada vez mais desatualizada e por sua vez menos utilizada [67].

2.4 SÍNTESE

Neste capitulo foi apresentado o ITS, a sua origem, o porquê da sua existência e a sua utilização. Foram abordados conceitos específicos do universo do ITS, como equipamentos que fazem parte das comunicações, tais como, o RSU e o OBU, as comunicações V2V e V2I e foram mencionados alguns protocolos de mensagens. Foram abordadas várias organizações que trabalham para melhorar a segurança de aplicações. Também foram apresentadas as metodologias de *penetration testing* mais adequadas ao projeto.

No próximo capítulo (3), é detalhado o papel fundamental do ITS no contexto do projeto e como foi aplicado. O capítulo ainda inclui uma análise comparativa das principais organizações de segurança no processo de seleção da abordagem mais adequada para o projeto. Além disso, é abordada a análise das metodologias de PenTest, revelando a escolha da metodologia considerada mais eficaz para avaliar e fortalecer a segurança do projeto.

3

ANÁLISE

No capítulo de análise, são explorados as três áreas essenciais que fundamentam o trabalho desenvolvido: ITS, Desenvolvimento Seguro e PenTest. O ITS representa a base tecnológica funcional do trabalho. Na esfera do desenvolvimento seguro, são comparadas organizações, tais como, OWASP, SANS, CIS *Controls* e CWE para incorporar as melhores práticas de segurança. Por fim, no PenTest, são avaliadas metodologias como OSSTMM, OWASP WSTG e PTES. Essa análise estabelece os fundamentos para o trabalho desenvolvido.

3.1 ITS

O ITS é um sistema que utiliza tecnologias de informação e comunicação para melhorar a gestão dos transportes públicos e privados e tornar mais eficiente a utilização dos recursos rodoviários. Como referido anteriormente, na secção 2.1, o ITS suporta vários protocolos de comunicação para facilitar a interconexão de diferentes dispositivos e sistemas.

Foi necessário criar um sistema informático para gerir os RSU e os sinais rodoviários. Para isso foi escolhido desenvolver uma plataforma *web* que possibilita aos utilizadores criar esses tipos de componentes intuitivamente na aplicação, sem perceberem que utilizam a tecnlogia ITS.

A plataforma desempenha um papel crítico na criação e gestão de sinais rodoviários e RSUs ao armazenar informações vitais. Esses dados incluem coordenadas geográficas, que indicam a localização exata de cada elemento no mapa da aplicação, a referência do equipamento, o alcance do emissor e uma descrição do RSU. Estes são dados essenciais para garantir que a plataforma possa monitorizar, rastrear e gerir eficazmente os RSU.

Esses dados são fundamentais para a tomada de decisões informadas e eficazes na gestão dos RSU no terreno. A precisão e confiabilidade destas informações desempenham um papel importante para garantir que os RSU sejam implantados e operados conforme o planeado.

A segurança dos dados no ITS é crucial, e qualquer alteração não autorizada ou roubo de informações podem ter sérias consequências. No contexto dos RSU,

manipulações indevidas podem comprometer a precisão das informações, levando a decisões erradas na gestão do tráfego. A integridade dos dados, como coordenadas geográficas e referências de equipamentos, é essencial para a eficácia do sistema.

Além disso, a conformidade com *standards*, como a ISO 2217 e a estrutura da mensagem IVIM para sinais rodoviários é crucial, e qualquer adulteração pode afetar a interoperabilidade e a qualidade das informações. Proteger contra possíveis alterações ou roubos de dados é fundamental para manter a eficiência e a segurança do sistema de transporte.

Os sinais rodoviários inseridos na aplicação, carecem das mesmas necessidades de precisão dos dados. Com a ajuda do *standard* ISO 2217 e da estrutura da mensagem IVIM, é possível identificar os requisitos que os sinais rodoviários têm para serem implementados na aplicação de forma a gerar informações adequadas.

A estrutura da mensagem IVIM, composta por um cabeçalho (*header*) e vários *containers*, é uma parte essencial neste projeto. Embora nem todos os *containers* sejam utilizados na aplicação *web*, os *containers* GLC, *General IVI Container* (GIC) e *Traffic Class* (TC) desempenham um papel crítico na criação e armazenamento de dados relacionados aos sinais rodoviários.

O GLC armazena informações sobre coordenadas do ponto de referência do sinal e os pontos que constituem as zonas. O GIC é responsável por armazenar informações das zonas do sinal, qual o *standard* utilizado (ISO ou Vienna) e no caso de ser um sinal ISO este *container* ainda armazena outros atributos associados a esse sinal. Por fim, o TC regista todas as informações textuais do sinal, como título e comentários.

Esta estrutura IVIM permite que a plataforma faça a leitura e interpretação dos sinais, automatizando a recolha de dados sempre que possível. Isso não só agiliza o processo de criação de sinais, mas também ajuda a planear a implantação dos RSU no terreno, permitindo uma cobertura eficiente para todos os sinais.

Cada sinal rodoviário inclui três zonas, *Awareness*, *Detection* e *Relevance*. Cada zona desempenha um papel específico na interação entre os sinais e os veículos nas estradas.

A *Awareness Zone* identifica o sinal pela primeira vez para o carro saber a existência do sinal. A *Detection Zone* informa o condutor sobre a presença iminente do sinal. Por fim, a *Relevance Zone* é onde o sinal efetivamente atua e onde o condutor tem de cumprir a regra de trânsito correspondente. A Figura 11 representa as zonas descritas previamente.

A plataforma permite que os sinais sejam inseridos em um de dois formatos de dados geográficos: *delta* e *absolute*.

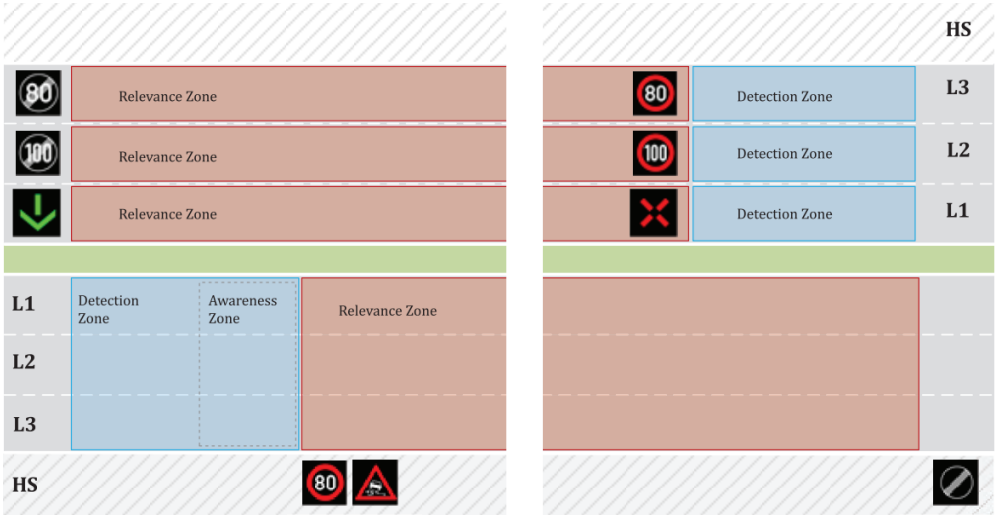


Figura 11: Zonas (fonte ISO21217 [9])

Os dados geográficos no formato *delta* representam valores correspondentes à diferença entre duas coordenadas geográficas. Os dados geográficos *absolute* fornecem coordenadas geográficas absolutas.

A plataforma também suporta dois *standards* de identificação de sinais (Vienna e ISO). A escolha do *standard* afeta como os dados são estruturados e compartilhados entre dispositivos e RSU, sendo crucial para a interoperabilidade e a comunicação eficaz entre os sistemas.

Os sinais ISO podem incluir atributos que fornecem detalhes relacionados aos veículos, como velocidade máxima ou mínima, peso máximo, dimensões máximas do veículo e categoria do sinal. Estes atributos são para importantes para garantir a correta interpretação de qualquer sinal.

A escolha das tecnologias Laravel para o *backend* e Vue.js para o *frontend* no desenvolvimento da plataforma já se encontrava definida aquando do início deste projeto. Estas tecnologias fornecem uma base sólida para o desenvolvimento de aplicações *web* robustas e interativas, garantindo eficiência, escalabilidade e uma experiência de utilizador avançada.

3.2 DESENVOLVIMENTO SEGURO

Ter um *website* inseguro pode desencadear diversos problemas, tanto para o proprietário do *website* quanto para os seus utilizadores [68]. A maioria das preocupações de segurança estão relacionadas a nível aplicacional. Um dos principais motivos prende-se pelo facto é das aplicações *web* serem acessíveis por navegadores e poderem ser acedidas por qualquer pessoa em qualquer lugar [69].

Alguns dos problemas em que estão associados a *websites* inseguros são:

- Roubo de dados: se o *website* não estiver seguro, os *hackers* podem roubar informações dos utilizadores, como *passwords*, endereços eletrónicos, informações bancarias e outras informações confidenciais.
- Propagação de *malware*: se o *website* for invadido, os *hackers* podem usá-lo para espalhar *malwares* para os visitantes da aplicação *web*. Isso pode levar a problemas como roubo de informações, perda de dados e até danos nos dispositivos dos utilizadores.
- Comprometimento da reputação: se os utilizadores souberem que o *website* não é seguro, eles provavelmente não irão confiar nele e podem evitar de o utilizar. Isso prejudica a sua reputação e afeta negativamente o negócio.
- Penalidades de *Search Engine Optimization* (SEO): os navegadores, como o Google, tendem a penalizar *websites* que são considerados inseguros ou que não seguem as melhores práticas de segurança. Isso pode levar a uma redução no tráfego do site e, portanto, reduzir a visibilidade do negócio que é suportado pela aplicação *web*.
- Problemas legais: dependendo do tipo de site e das leis em vigor no seu país, o proprietário da aplicação *web* pode ser responsabilizado se pelo menos um utilizador sofrer danos devido à falta de segurança no *website*, como dados expostos de clientes que não estejam conforme o Regulamento Geral de Proteção de Dados (RGPD).

Ter um *website* inseguro pode ter consequências sérias e prejudiciais para as organizações e para os seus utilizadores. É essencial garantir que as aplicações *web* estejam sempre protegidas e atualizadas com as melhores práticas de segurança para manter a segurança dos seus dados e dos seus visitantes. Assim como em cumprimento com as leis em vigor para evitar problemas judiciais.

3.2.1 *Frameworks de segurança*

Devido à importância e à dificuldade de garantir a segurança nas várias aplicações e nas suas organizações, várias instituições decidiram criar diversas *frameworks* de segurança. Desta forma existem *frameworks* para os mais diferentes cenários, desde a *framework Health Insurance Portability and Accountability Act* (HIPPA) que serva para proteger as informações de saúde dos pacientes ou a NIST *Cybersecurity Framework* desenvolvida para identificação abrangente e personalizada de vulnerabilidades de segurança, tanto a nível aplicacional como organizacional.

Por consequência do grande número de *frameworks* de segurança existentes é aconselhável escolher sempre a mais adequada aos requisitos da organização ou da aplicação. Isto porque existem *frameworks* redirecionadas às organizações, aplicações ou apenas a uma funcionalidade específica de *software*. No caso deste projeto a única preocupação é desenvolver uma aplicação *web* segura, sem preocupar com a organização onde ele atuará.

As *frameworks* de segurança referidas e abordadas no capítulo 2 foram escolhidas após várias pesquisas para perceber a melhor forma de desenvolver um *webiste* seguro. No contexto específico de desenvolvimento de aplicações *web* não existem muitas *frameworks* de segurança.

Embora cada uma dessas *frameworks* de segurança tenha as suas próprias características e foco, todas são valiosas para a segurança cibernética. O OWASP *top* Ten fornece uma visão geral dos principais riscos de segurança em aplicações *web*, enquanto o SANS e o CIS Controls oferecem orientações e diretrizes sobre como implementar controlos de segurança eficazes. O CWE é uma ferramenta útil para identificar e classificar vulnerabilidades de segurança de *software* e ajuda os programadores a se concentrarem em áreas específicas para melhorar a segurança dos seus sistemas.

Neste projeto não foi escolhida apenas uma *framework* de segurança, dado que o objetivo principal ser o desenvolvimento de uma aplicação *web* segura. Este projeto serviu ainda para explorar algumas das *frameworks* disponíveis no mercado. Foi utilizado o OWASP *top* TEN como ferramenta principal para o desenvolvimento da aplicação com o *top* 25 de erros de *software* mais perigosos, segundo o CWE e o SANS. Das 25 vulnerabilidades do CWE, apenas foram consideradas 14.

O CIS Controls foi a única *framework* mencionada que não foi utilizada. Esta *framework* de segurança teêm vários controlos que não têm interesse para o contexto do projeto. Obviamente que a *framework* permite apenas a utilização dos controlos que sejam pertinentes para o utilizador, no entanto, não existe essa necessidade visto que as restantes *frameworks* abrangem de forma geral as vulnerabilidades descritas nos restantes controlos do CIS.

3.2.2 *Frameworks escolhidas*

A CWE *top* 25 é uma lista de vulnerabilidades de segurança de *software* encontradas em toda a indústria de desenvolvimento de *software*. É um projeto voltado para a comunidade, desenvolvido pela *Massachusetts Institute of Technology Research and Engineering* (MITRE) em parceria com o SANS. Para cada entrada, o CWE fornece uma descrição da vulnerabilidade e as etapas para mitigá-la.

O projeto *top* 10 do OWASP têm uma lista semelhante de compilação de vulnerabilidades de *software* dirigida pela comunidade, mas direcionada exclusivamente para *websites*. Embora o CWE *top* 25 e o OWASP *top* 10 sejam diferentes, eles compartilham muitas das mesmas vulnerabilidades, permitindo a utilização de ambas. Na tabela 1 é possível identificar algumas das diferenças e semelhanças das duas *frameworks*.

Semelhanças	Diferenças
Ambas as listas destacam vulnerabilidades comuns e conhecidas em *software*, que podem ser exploradas por atacantes para comprometer a segurança do sistema.	A lista OWASP *top* 10 é mais focada em vulnerabilidades específicas relacionadas à segurança de aplicações *web*, enquanto a lista CWE *top* 25 é mais abrangente, incluindo vulnerabilidades que podem ocorrer em diferentes tipos de *software*.
Ambas as listas incluem vulnerabilidades de diversas categorias em comum, como injeção, autenticação e autorização inadequadas, controlo de acesso, problemas de configuração e exposição de informações sensíveis.	A lista CWE *top* 25 utiliza um esquema de pontuação para avaliar a gravidade das vulnerabilidades, enquanto a lista OWASP *top* 10 não. A pontuação da CWE é baseada em vários fatores, como a probabilidade de exploração bem-sucedida da vulnerabilidade e o impacto que ela teria se fosse explorada.
Ambas as listas enfatizam a importância da prevenção e mitigação dessas vulnerabilidades desde o início do ciclo de vida do desenvolvimento de *software*.	As listas têm algumas vulnerabilidades em comum, como injeção de SQL e XSS, mas também têm algumas diferenças, como a inclusão da CWE-787 *Out-of-bounds Write* e CWE-416 *Use After Free* na lista CWE *top* 25, entre outras, que não estão presentes na lista OWASP *top* 10.

Tabela 1: Semelhanças e diferenças do OWASP *top* 10 e do CWE *top* 25

Em resumo, tanto o OWASP *top* 10 quanto o CWE *top* 25 pretendem auxiliar os programadores a identificar e mitigar vulnerabilidades comuns nos seus sistemas, mas a lista do CWE é mais ampla e geral. A lista do OWASP *top* 10 é mais focada em vulnerabilidades específicas comuns em aplicações *web*.

A lista do OWASP é composta por dez principais vulnerabilidades de segurança em aplicações *web* e serve como um guia para programadores, *testers*, gestores de projetos e outros profissionais de TI. De modo a identificar, avaliar e corrigir essas vulnerabilidades. A lista mais recente é de 2021 e é possível visualizar na Figura 12 o *top* 10 [31].

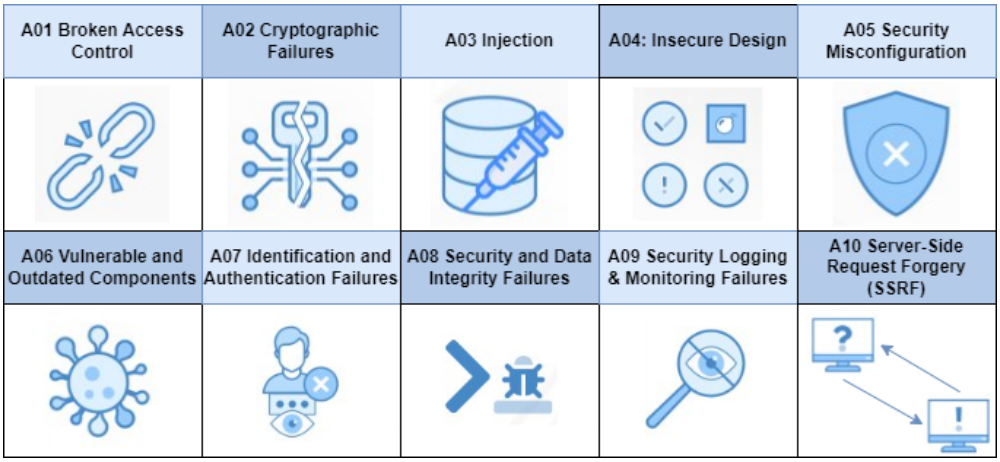


Figura 12: OWASP *top* 10 – 2021 (adaptado de [70])

A CWE *top* 25 é uma lista das 25 principais vulnerabilidades de *software* mais perigosas e é um sistema de classificação de vulnerabilidades de *software* que visa ajudar a identificar, compreender e prevenir vulnerabilidades em *software*. A lista da CWE *top* 25 é organizada em ordem de severidade, com a vulnerabilidade mais perigosa em primeiro lugar. Na tabela 2 está representada a lista do CWE de 2022, sendo esta a versão mais recente [71].

Rank	ID	Nome
1	CWE-787	*Out-of-bounds Write*
2	CWE-79	*Improper Neutralization of Input During Web Page Generation (XSS)*
3	CWE-89	*Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')*
4	CWE-20	*Improper Input Validation*
5	CWE-125	*Out-of-bounds Read*
6	CWE-78	*Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')*
7	CWE-416	*Use After Free*
8	CWE-22	*Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')*

Rank	ID	Nome
9	CWE-352	CSRF
10	CWE-434	*Unrestricted Upload of File with Dangerous Type*
11	CWE-476	*NULL Pointer Dereference*
12	CWE-502	*Deserialization of Untrusted Data*
13	CWE-190	*Integer Overflow or Wraparound*
14	CWE-287	*Improper Authentication*
15	CWE-798	*Use of Hard-coded Credentials*
16	CWE-862	*Missing Authorization*
17	CWE-77	*Improper Neutralization of Special Elements used in a Command ('Command Injection')*
18	CWE-306	*Missing Authentication for Critical Function*
19	CWE-119	*Improper Restriction of Operations within the Bounds of a Memory Buffer*
20	CWE-276	*Incorrect Default Permissions*
21	CWE-918	SSRF
22	CWE-362	*Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')*
23	CWE-400	*Uncontrolled Resource Consumption*
24	CWE-611	*Improper Restriction of XML External Entity Reference*
25	CWE-94	*Improper Control of Generation of Code ('Code Injection')*

Tabela 2: CWE *top* 25 das vulnerabilidades (adaptado de [71])

3.2.3 *União das frameworks*

Ambas as listas destacam vulnerabilidades comuns e conhecidas em *software*. Apesar de o CWE não ser direcionado para aplicações *web* têm muitas vulnerabilidades em comum ao OWASP. Algumas vulnerabilidades do CWE podem ser enquadradas nas várias vulnerabilidades do OWASP, conforme apresentado na Figura 13.

Como é possível observar na Figura 13, nem todas as vulnerabilidades do *top* 25 do CWE podem ser associadas ao *top* 10 do OWASP. Para aumentar o contexto entre as duas *frameworks* foram associadas vulnerabilidades que pertencem ao *top* 40 do

CWE com o auxílio da documentação oficial do OWASP. Para cada vulnerabilidade do *top* 10 do OWASP, o OWASP menciona todas as CWE associadas a essa vulnerabilidade. O que auxiliou bastante esta análise comparativa.

OWASP A01:2021 Broken Access Control	**OWASP A02:2021 Cryptographic Failures**
(Rank 8) CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') **(Rank 20)** CWE-276: Incorrect Default Permissions **(Rank 9)** CWE-352: Cross-Site Request Forgery (CSRF) **(Rank 35)** CWE-601: URL Redirection to Untrusted Site ('Open Redirect') **(Rank 32)** CWE-668: Exposure of Resource to Wrong Sphere **(Rank 16)** CWE-862: Missing Authorization **(Rank 28)** CWE-863: Incorrect Authorization	**(Rank 39)** CWE-319: Cleartext Transmission of Sensitive Information
OWASP A03:2021 Injection	**OWASP A04:2021 Insecure Design**
(Rank 4) CWE-20: Improper Input Validation **(Rank 17)** CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection') **(Rank 6)** CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') **(Rank 2)** CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') **(Rank 3)** CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') **(Rank 25)** CWE-94: Improper Control of Generation of Code ('Code Injection')	**(Rank 29)** CWE-269: Improper Privilege Management **(Rank 40)** CWE-312: Cleartext Storage of Sensitive Information **(Rank 10)** CWE-434: Unrestricted Upload of File with Dangerous Type **(Rank 38)** CWE-522: Insufficiently Protected Credentials
OWASP A05:2021 Security Misconfiguration	**OWASP A06:2021 Vulnerable and Outdated Components**
(Rank 24) CWE-611: Improper Restriction of XML External Entity Reference	Nenhuma CWE correspondente ao Top 40
OWASP A07:2021 Identification and Authentication Failures	**OWASP A08:2021 Software and Data Integrity Failures**
(Rank 14) CWE-287: Improper Authentication **(Rank 26)** CWE-295: Improper Certificate Validation **(Rank 18)** CWE-306: Missing Authentication for Critical Function **(Rank 15)** CWE-798: Use of Hard-coded Credentials	**(Rank 12)** CWE-502: Deserialization of Untrusted Data
OWASP A09:2021 Security Logging and Monitoring Failures	**OWASP A10:2021 Server-Side Request Forgery (SSRF)**
Nenhuma CWE correspondente ao Top 40	**(Rank 21)** CWE-918: Server-Side Request Forgery (SSRF)

Figura 13: Associação do OWASP *top* 10 com o CWE *top* 40

Existem vários *blogs* [72] e artigos [73], que põem em causa o *top* 10 do OWASP por este abordar apenas 10 vulnerabilidades, existindo muitas vulnerabilidades para além do *top* 10. Neste projeto é apenas utilizado o *top* 10, para testar se é suficiente ou não para desenvolver uma aplicação *web* segura, visto que posteriormente é realizado um PenTest para verificar a segurança da aplicação *web*. É de notar que

independentemente do resultado, é sempre melhor utilizar alguma ferramenta de segurança, como o OWASP do que não utilizar.

O OWASP pode servir para dar início à introdução da segurança em aplicações *web* e, conforme a necessidade das organizações, estas podem aumentar a segurança das suas aplicações. Isto é um bom cenário para empresas que ainda não protegem as suas aplicações regularmente e estão apenas a começar o processo de implementação de protocolos internos de segurança. Desta forma, as suas aplicações estão protegidas contras as 10 vulnerabilidades mais perigosas, segundo o OWASP.

3.2.4 *Broken Access Control*

O *Broken Access Control* é uma vulnerabilidade de segurança em aplicações *web* que permite a um utilizador mal-intencionado aceder a recursos ou funcionalidades que deveriam ser restritas.

Esta vulnerabilidade ocorre quando a aplicação *web* não implementa corretamente as políticas de controlo de acesso, permitindo que utilizadores não autorizados acedam a recursos ou funcionalidades que deveriam estar disponíveis apenas para utilizadores com permissão adequada. Isto pode ocorrer devido a falhas no processo de autenticação, autorização ou validação de entrada de dados.

Dos dados fornecidos ao OWASP, este testou 94.55% de aplicações *web* com algum ataque do tipo de *broken access control*, com uma taxa máxima de incidência de 55.97%, uma taxa média de incidência de 3.81% e 318 mil ocorrências. Esta vulnerabilidade enquadra um total de 34 vulnerabilidades do CWE [74].

3.2.4.1 *Vulnerabilidades*

O controlo de acesso implementa uma política de segurança de forma que os utilizadores não possam agir fora das permissões atribuídas. As falhas geralmente levam à divulgação não autorizada de informações, modificação ou destruição de todos os dados [74]. Vulnerabilidades comuns de controlo de acesso incluem:

- Violação do princípio de privilégio mínimo ou negação por padrão, em que o acesso deve ser concedido apenas para recursos, funções ou utilizadores específicos, mas está disponível para qualquer pessoa.
- Ignorar as verificações de controlo de acesso modificando a *Uniform Resource Locator* (URL) (violação de parâmetros ou navegação forçada), o estado interno da aplicação ou a página *HyperText Markup Language* (HTML), ou utilizando uma ferramenta de ataque que modifica solicitações de *Application Programming Interface* (API).

- Permitir visualizar ou editar a conta de outra pessoa, fornecendo o seu identificador exclusivo (referências diretas inseguras de objetos).
- Ao aceder à API com controlos de acesso para *POST*, *PUT* e *DELETE*.
- Elevação de privilégio. Atuar como utilizador sem estar conectado ou atuar como administrador quando conectado como utilizador.
- Manipulação de metadados, como reprodução ou adulteração de um *token* de controlo de acesso *JSON Web Token* (JWT), ou um *cookie* ou campo oculto manipulado para elevar privilégios ou abusar da invalidação de JWT.
- A configuração incorreta do *Cross-Origin Resource Sharing* (CORS) permite acesso à API de origens não autorizadas ou não confiáveis.
- A navegação em páginas que exigem autenticação como utilizador não autenticado.

3.2.4.2 *Possíveis consequências*

O impacto de um invasor ter sucesso num ataque de *Broken Access Control* pode ser bastante significativo e variado, dependendo do tipo de recurso ou funcionalidade que o invasor consegue empossar indevidamente. Algumas possíveis consequências de um ataque bem-sucedido incluem:

- Acesso a informações confidenciais: O invasor pode ter acesso a informações confidenciais, como dados pessoais dos clientes, informações financeiras ou segredos comerciais.
- Modificação ou exclusão de dados: O invasor pode modificar ou excluir dados importantes, o que pode levar a perda de informações críticas ou comprometer a integridade dos dados armazenados.
- Execução de ações não autorizadas: O invasor pode executar ações não autorizadas, como transferências de fundos, alterações em configurações críticas ou inclusão de conteúdo malicioso.
- Prejuízos financeiros: Um ataque bem-sucedido de *Broken Access Control* pode levar a prejuízos financeiros significativos para a organização, seja por meio de multas ou processos legais, ou pela perda de clientes, ou oportunidades de negócios.
- Danos à reputação: A revelação de uma falha de segurança pode ter um impacto negativo na reputação da organização, levando a perda de confiança dos utilizadores, clientes ou investidores.

3.2.4.3 *Como prevenir*

O controlo de acesso só é eficaz em código confiável do lado do servidor, no qual o invasor não pode modificar a verificação ou os metadados do controlo de acesso:

- Exceto para recursos públicos, negados por padrão.
- Implementar mecanismos de controlo de acesso uma vez e reutilize-os em toda a aplicação, incluindo a minimização do uso de CORS.
- Os controlos de acesso do modelo devem impor a propriedade do registo em vez de aceitar que o utilizador possa criar, ler, atualizar ou excluir qualquer registo.
- Os requisitos exclusivos de limite de negócios da aplicação devem ser impostos pelos modelos de domínio.
- Desabilitar a lista de diretórios do servidor *web* e certificar que os metadados do ficheiro (por exemplo, ".git") e os ficheiros de *backup* não estejam presentes nas raízes da aplicação *web*.
- Registe as falhas de controlo de acesso, alerte os administradores quando apropriado (por exemplo, falhas repetidas).
- API de limite de taxa e acesso ao controlador para minimizar os danos causados por ferramentas de ataque automatizadas.
- Os identificadores de sessão com estado devem ser invalidados no servidor após o *logout*. Os *tokens* JWT sem estado devem ter vida curta para que a janela de oportunidade para um invasor seja minimizada. Para JWT de longa duração, é altamente recomendável seguir os padrões *Open Authorization* (OAuth) para revogar o acesso.

Os programadores e as equipas de controlo de qualidade devem incluir a unidade de controlo de acesso funcional e os testes de integração.

3.2.5 *Cryptographic Failures*

Um ataque de *Cryptographic Failures* em aplicações *web* ocorre quando a criptografia utilizada na transmissão ou armazenamento de dados é mal implementada ou insegura. Isso pode levar a falhas de segurança, como a divulgação de informações sensíveis ou a falsificação de dados.

Dos dados enviados para o OWASP, foram testados 79.33% dos *websites* com algum tipo de ataque de falhas de criptografia, com uma taxa máxima de incidência

de 46.44%, uma taxa média de incidência de 4.49% e um total de 233 mil ocorrências. Esta vulnerabilidade enquadra um total de 29 vulnerabilidades do CWE [75].

3.2.5.1 *Vulnerabilidades*

É fundamental determinar as necessidades de proteção dos dados em trânsito e em repouso. Por exemplo, *passwords*, números de cartões de crédito, registos de saúde, informações pessoais e segredos comerciais exigem proteção extra, principalmente se esses dados estiverem sob leis de privacidade, por exemplo, RGPD [75]. Para todos esses dados:

- Verificar todo o tráfego interno, por exemplo, entre *load balancers* servidores *web* ou sistemas de *back-end*.
- Ter atenção a algoritmos ou protocolos criptográficos antigos, ou fracos por padrão, ou em código mais antigo.
- Validar devidamente o certificado do servidor.
- Analisar se a criptografia é aplicada, por exemplo, verificar a segurança de cabeçalho *HyperText Transfer Protocol* (HTTP) (*browser*) ou cabeçalhos em falta.
- Não utilizar funções de algoritmo de *hash* obsoletas, como *Message-Digest* (MD5) ou *Secure Hash Algorithm 1* (SHA1).
- Não utilizar métodos de preenchimento criptográfico obsoletos, como *Public Key Cryptography Standards 1* (PKCS).
- Não utilizar mensagens de erro criptográficas ou informações do canal lateral porque estas são exploráveis, por exemplo, na forma de ataques de preenchimento.

3.2.5.2 *Possíveis consequências*

Um ataque do tipo *Cryptographic Failures* a um *website* se for bem-sucedido, pode comprometer a segurança dos dados e informações confidenciais dos utilizadores, como senhas, números de cartões de crédito e informações pessoais. Por sua vez, isto pode levar a roubo de identidade, fraude financeira, perda de reputação da organização, processos judiciais e violações de regulamentações de proteção de dados.

Além disso, se um invasor puder aceder e modificar os dados em trânsito ou em repouso, pode resultar em perda de integridade dos dados, que por sua vez pode levar a decisões incorretas baseadas em informações imprecisas ou incompletas.

3.2.5.3 *Como prevenir*

É importante que as empresas e organizações protejam os seus sistemas e aplicações *web* contra ataques direcionados a *Cryptographic Failures*, implementando medidas de segurança adequadas, como criptografia forte, autenticação forte, gestão de chaves seguras e práticas de desenvolvimento seguro de *software*. O OWASP fornece algumas medidas a serem tomadas para prevenir ataques *Cryptographic Failures*:

- Classificar os dados processados, armazenados ou transmitidos pela aplicação e identificar os dados sensíveis conforme as leis de privacidade, regulamentações ou necessidades empresariais.
- Não armazenar dados sensíveis desnecessariamente. Descartá-los o mais cedo possível ou utilizar *tokens* compatíveis com *Payment Card Industry Data Security Standard* (PCI DSS) ou *truncation*. Dados não retidos não podem ser roubados.
- Certificar que todos os dados sensíveis em repouso estão encriptados.
- Garantir que algoritmos, protocolos e chaves pré-definidas estejam atualizados e que sejam fortes, assim como utilizar uma gestão adequada de chaves.
- Desabilitar a *cache* para respostas que contenham dados sensíveis.
- Aplicar controlos de segurança necessários conforme a classificação de dados.
- Armazenar senhas utilizando funções de *hashing* adaptáveis, *salted* e com um fator de trabalho (*delay factor*) forte, como *Argon2*, *Bcrypt* ou *Password-Based Key Derivation Function* (PBKDF).
- Evitar funções criptográficas e esquemas de preenchimento depreciados, como MD5, SHA1, PKCS.

A utilização de *frameworks* de desenvolvimento ajudam com algumas destas medidas, como o Laravel com recursos de criptografia e de proteção de rotas.

3.2.6 *Injection*

Um ataque de injeção em aplicações *web* é uma técnica de exploração de vulnerabilidades na qual o invasor insere código malicioso num *website* para fazer com que ele execute ações inesperadas ou inseguras. O tipo mais comum de ataque de injeção em aplicações *web* é a injeção de *Structured Query Language* (SQL), que envolve a inserção de comandos SQL maliciosos em formulários de entrada ou outros campos que interagem com a base de dados.

Dos dados fornecidos ao OWASP, este testou 94.04% de aplicações *web* com algum ataque do tipo de injeção, com uma taxa máxima de incidência de 19.09%, uma taxa média de incidência de 3.37% e 274 mil ocorrências. Esta vulnerabilidade enquadra um total de 33 vulnerabilidades do CWE [76].

3.2.6.1 *Vulnerabilidades*

Os ataques de injeção podem ser evitados validando adequadamente a entrada de dados, utilizando parâmetros preparados ou variáveis de ligação em consultas à base de dados e restringindo o acesso de utilizadores aos recursos do sistema. É importante que os programadores de *websites* entendam essas técnicas de ataque e tomem medidas para proteger as aplicações *web*.

A aplicação é vulnerável a um ataque de *injection* quando:

- Os dados fornecidos pelo utilizador não são validados, filtrados ou limpos pela aplicação.
- Consultas dinâmicas ou chamadas não parametrizadas sem escape sensível ao contexto são utilizados diretamente ao interpretador.
- Os dados inseridos pelo invasor são utilizados nos parâmetros de pesquisa de *Object-Relational Mapping* (ORM) para extrair registos confidenciais adicionais.
- Dados inseridos pelo invasor são utilizados diretamente ou concatenados. O SQL ou comando contêm a estrutura e os dados maliciosos em consultas dinâmicas, comandos ou procedimentos armazenados.

Algumas das injeções mais comuns são SQL, *Not only Structured Query Language* (NoSQL), comandos do sistema operativo, ORM, *Lightweight Directory Access Protocol* (LDAP) e injeção de linguagem de expressão ou *Object-Graph Navigation Language* (OGNL). O conceito é idêntico entre todos os intérpretes, a revisão do código-fonte é o melhor método para detetar se as aplicações são vulneráveis a injeções.

Testes automatizados em todos os parâmetros, cabeçalhos, URL, *cookies*, *JavaScript Object Notation* (JSON), *Simple Object Access Protocol* (SOAP) e entradas de dados *Extensible Markup Language* (XML) são fortemente encorajados. As organizações podem incluir ferramentas de *Static Application Security Testing* (SAST), *Dynamic Application Security Testing* (DAST) e *Interactive Application Security Testing* (IAST) no *pipeline* de *Continuous integration/Continuous delivery* (CI/CD) para identificar falhas de injeção introduzidas antes entrada em produção da aplicação [76].

3.2.6.2 *Possíveis consequências*

No caso de o invasor ter sucesso num ataque de injeção no *website*, as consequências podem ser muito graves e variadas. Isso ocorre porque um ataque de injeção permite que o invasor injete código malicioso no sistema e execute ações não autorizadas. Alguns dos possíveis impactos de um ataque de injeção são:

- Roubo de dados sensíveis: O invasor pode obter acesso não autorizado a dados confidenciais, como informações pessoais, senhas, informações financeiras ou de negócios.
- Comprometimento do sistema: O invasor pode assumir o controlo total do sistema ou infetá-lo com *malware*, permitindo que ele execute ações maliciosas em larga escala.
- Destruição de dados: O invasor pode destruir ou corromper dados importantes do sistema, causando perda de dados significativa.
- Ataques secundários: O invasor pode utilizar a vulnerabilidade de injeção para realizar ataques secundários, como instalar *backdoors* ou explorar outras vulnerabilidades.
- Perda de reputação: Se dados confidenciais forem expostos ou a aplicação *web* ficar indisponível, a reputação da organização que possui o *website* pode ser prejudicada, tal como o seu negócio.

3.2.6.3 *Como prevenir*

Para prevenir ataques de injeção, é necessário codificar os *websites* com segurança para não poderem ser facilmente manipulados. Dependendo da natureza das aplicações *web*, existem diversas técnicas que podem ser implementadas para prevenir ataques.

- Validar entradas do utilizador: Os dados inseridos pelos utilizadores são indicados como a chave de ataques de injeção. Permitir o sistema aberto para receber todos os dados enviados pelos utilizadores é um convite para o desastre. Filtrar dados por contexto também é eficaz ao escolher as entradas apropriadas em várias situações. Por exemplo, no caso de endereços de correio eletrónico, deve ser permitido apenas caracteres e números. Para números de telefone, deve ser permitido apenas números.
- Limitar acesso a privilégios essenciais: O grau de danos que um invasor pode infligir depende do nível de acesso que ele tem. Deve ser restrito o acesso dos utilizadores, não permitindo privilégios de administração nos sistemas que utilizam para conectar à base de dados. Os utilizadores devem ter acesso

limitado para a maioria das suas atividades, desta forma, se um *hacker* obtiver acesso à conta, haverá um limite para o que ele pode fazer.

- Atualizar e corrigir: As aplicações *web* são mais propensas a ataques de injeção quando o *software* está desatualizado. Os invasores estão sempre à procura por vulnerabilidades para explorar.
- Proteger informações sensíveis: É necessário ter em atenção as informações exibidas no *website*. Por exemplo, uma mensagem de erro que parece inofensiva pode dar a um invasor uma oportunidade para penetrar o sistema. Garantir que quaisquer mensagens exibidas na aplicação *web* não contenham informações vitais. Por exemplo, quando o utilizador falha o processo de autenticação nunca se deve dizer que é a senha que está errada, isto porque o utilizador assim consegue descobrir quais são os endereços eletrónicos existentes na base de dados e posteriormente utilizar *brute force* para tentar descobrir a senha.
- Utilizar uma *Web Application Firewall* (WAF): Utilizar *firewalls* de aplicação *web* confiáveis ajuda a bloquear entradas de utilizadores ou solicitações de dados suspeitas. Os recursos de segurança das *firewalls* mais recentes são fortes o suficiente para detetar ameaças emergentes mesmo antes de uma correção ser fornecida.

Para além das preocupações descritas anteriormente é de notar que para prevenir a injeção é necessário manter os dados separados dos comandos e consultas. A opção preferencial é utilizar uma API segura, que evita totalmente o uso do interpretador, fornece uma *interface* parametrizada ou migra para ORM.

Para quaisquer consultas dinâmicas residuais, escape de caracteres especiais utilizando a sintaxe de escape específica para esse interpretador.

3.2.7 *Insecure Design*

Um ataque de *Insecure Design* em aplicações *web* ocorre quando a aplicação é desenvolvida com falhas de segurança no seu *design*, permitindo que um invasor possa explorar essas vulnerabilidades para comprometer a integridade, confidencialidade ou disponibilidade dos dados e sistemas envolvidos. Essas falhas podem incluir a falta de autenticação adequada, validação inadequada de entrada de dados, ou a exposição excessiva de informações sensíveis.

A análise dos dados enviados para o OWASP, resultou em 77.25% *websites* testados com algum tipo de ataque direcionado para *Insecure Design*, obteve uma taxa máxima de incidência de 24.19%, uma taxa média de incidência de 3% e

um total de 262 mil ocorrências. Esta vulnerabilidade enquadra um total de 40 vulnerabilidades do CWE.

3.2.7.1 *Vulnerabilidades*

Insecure Design é uma categoria ampla que representa diferentes fraquezas, expressas como "*design* de controlo em falta ou ineficaz". Insegurança de *design* não é a fonte para todas as outras categorias de risco *top* 10. Há uma diferença entre insegurança de *design* e implementação insegura. O OWASP difere entre falhas de *design* e defeitos de implementação por uma razão, eles têm causas raiz e remediação diferentes. Um *design* seguro ainda pode ter defeitos de implementação levando a vulnerabilidades que podem ser exploradas.

Um *design* inseguro não pode ser corrigido por uma implementação perfeita, já que, por definição, controlos de segurança necessários nunca foram criados para proteção contra ataques específicos. Um dos fatores que contribuem para a insegurança de *design* é a falta de perfil de risco empresarial inerente ao *software* ou sistema em desenvolvimento e, portanto, a falha em determinar o nível de *design* de segurança necessário.

Gestão de requisitos e recursos

São necessárias reuniões e negociações para reunir os requisitos de negócios para uma aplicação, incluindo requisitos de proteção relacionados à confidencialidade, integridade, disponibilidade e autenticidade de todos os ativos de dados e a lógica de negócio esperada.

***Design* seguro**

Design seguro é uma cultura e metodologia que constantemente avalia ameaças e garante que o código seja projetado e testado robustamente para evitar métodos de ataque conhecidos. O *design* seguro não é um complemento nem uma ferramenta que pode ser adicionada ao *software*, mas sim uma cultura e metodologia incorporadas ao ciclo de desenvolvimento.

Ciclo de vida do desenvolvimento seguro

Software seguro requer um ciclo de vida de desenvolvimento seguro, algum tipo de padrão de *design* seguro, metodologia de estrada pavimentada, biblioteca de componentes seguros, ferramentas e modelagem de ameaças.

3.2.7.2 *Possíveis consequências*

Um ataque bem-sucedido de *insecure design* pode ter várias consequências graves para a segurança do sistema e dos dados. Algumas das principais consequências incluem:

- Comprometimento de dados sensíveis: Um *design* inseguro pode permitir que um invasor acede a dados confidenciais, como informações financeiras, dados pessoais ou de clientes, informações de saúde, entre outros.
- Exploração de vulnerabilidades: O *design* inseguro pode criar vulnerabilidades no sistema que podem ser exploradas por invasores para lançar ataques, como injeção de código, ataques de negação de serviço, escalonamento de privilégios, entre outros.
- Danos à reputação: Uma violação de dados ou outro ataque bem-sucedido pode prejudicar a reputação da empresa e afetar a confiança dos clientes.
- Interrupção dos negócios: Um ataque bem-sucedido pode resultar em interrupções significativas nos negócios, que podem levar a perda de produtividade, perda de rendimento e outros problemas.

Em geral, um ataque de *insecure design* pode ter consequências graves para uma organização e deve ser evitado por meio da implementação de práticas de segurança adequadas durante todo o ciclo de vida do desenvolvimento de *software*.

3.2.7.3 *Como prevenir*

Para prevenir ataques de *insecure design* em aplicações *web*, é preciso adotar uma cultura e metodologia de *design* seguro, integrando a modelagem de ameaças desde o início do projeto, avaliando constantemente as ameaças e garantindo que o código seja projetado e testado robustamente para evitar métodos de ataque conhecidos. Existem várias técnicas que podem ser implementadas para prevenir ataques em aplicações *web*, dependendo da sua natureza:

- Deve ser adotada uma cultura de *design* seguro desde o início do desenvolvimento de *software*.
- As ameaças potenciais para a aplicação devem ser identificadas e documentadas, criando estratégias para minimizar esses riscos.
- Os ataques anteriores devem servir para aprendizagem e deve ser implementado na abordagem de *design* seguro.
- Controlos de segurança robustos devem ser implementados em todos os níveis da aplicação, incluindo controlo de acesso, autenticação e validação de entrada.

- A aplicação deve ser atualizada regularmente com as correções necessárias, mantendo-se atualizado com as últimas vulnerabilidades e ameaças de segurança.

3.2.8 *Security Misconfiguration*

Um ataque de *Security Misconfiguration* em aplicações *web* ocorre quando as configurações de segurança da aplicação são inadequadas ou mal configuradas, permitindo que invasores explorem vulnerabilidades e acedem a informações sensíveis ou realizem ações maliciosas na aplicação. Isso pode incluir a exposição de informações confidenciais, a execução de código malicioso ou a alteração de dados na aplicação. É importante garantir que as configurações de segurança estejam corretamente configuradas e atualizadas para evitar esses tipos de ataques.

O OWASP testou um grande número de aplicações *web* e descobriu que cerca de 89,58% delas tinham alguma forma de vulnerabilidade de *Security Misconfiguration*. A taxa máxima de incidência foi de 19,84%, com uma taxa média de 4,51% e um total de 208 mil ocorrências. Esta vulnerabilidade do OWASP engloba 33 vulnerabilidades do CWE.

3.2.8.1 *Vulnerabilidades*

Alguns dos problemas mais comuns que podem deixar uma aplicação vulnerável a *Security Misconfiguration* incluem:

- A falta de medidas de segurança adequadas em qualquer parte da estrutura da aplicação ou nas permissões em serviços *cloud* mal configuradas.
- Funcionalidades desnecessárias não podem estar ativadas ou instaladas (por exemplo, portas, serviços, páginas, contas ou privilégios desnecessários).
- Contas de utilizadores e as suas palavras-passe predefinidas estarem ativas e não tenham sido alteradas.
- A gestão de erros não pode expor o caminho dos erros e as mensagens de erro excessivamente informativas aos utilizadores.
- As configurações de segurança nos servidores de aplicação, *frameworks* de aplicação (por exemplo, *Struts*, *Spring*, *ASP.NET*), bibliotecas, bases de dados, entre outros, devem estar definidas com valores seguros.
- O *software* deve estar atualizado ou não vulnerável (ver 3.2.9).

Se não houver um processo bem definido e repetitivo para configurar a segurança da aplicação, o risco de segurança para os sistemas será maior.

3.2.8.2 *Possíveis consequências*

As vulnerabilidades de *Security Misconfiguration* em aplicações *web* podem ter consequências graves para a segurança da aplicação e dos seus utilizadores. Alguns exemplos de possíveis consequências incluem:

- Exposição de informações sensíveis: Através da exploração de vulnerabilidades de configuração inadequada, os atacantes podem obter acesso a informações sensíveis, como dados de utilizadores ou informações financeiras.
- Execução de código malicioso: A exploração de vulnerabilidades de configuração malfeita pode permitir que os atacantes executem código malicioso na aplicação, abrindo a porta para ataques mais graves, como roubo de dados, infeção por *malware*, entre outros.
- Comprometimento da integridade da aplicação: As vulnerabilidades de *Security Misconfiguration* podem permitir que os atacantes manipulem a aplicação para comprometer a sua integridade e confiabilidade, por exemplo, alterando dados ou realizando transações fraudulentas.
- Danos à reputação: A descoberta de vulnerabilidades de configuração inadequada pode prejudicar a reputação da aplicação e da organização responsável, afetando a confiança dos utilizadores e a imagem da empresa.
- Sanções regulatórias e legais: Em muitos setores, a violação de requisitos regulatórios de segurança pode resultar em multas e outras sanções legais, além de danos à imagem e reputação da organização.

3.2.8.3 *Como prevenir*

A prevenção de vulnerabilidades de segurança é crucial para garantir a proteção adequada de uma aplicação *web*. Para prevenir ataques direcionados a *Security Misconfiguration*, é importante implementar processos de instalação seguros, tais como:

- Ambientes de desenvolvimento e produção devem ser configurados de forma idêntica, com diferentes credenciais utilizadas em cada ambiente. Esse processo deve ser automatizado para minimizar o esforço necessário para configurar um novo ambiente seguro.
- Utilizar uma plataforma mínima sem quaisquer recursos, componentes, documentação e amostras desnecessários. Remover ou não instalar recursos e *frameworks* não utilizados.

- Adotar uma arquitetura de aplicação segmentada que forneça separação efetiva e segura entre componentes ou inquilinos, com segmentação, *Access Control Lists* (ACLs) ou virtualização, como *containers*.

Seguindo estas medidas de prevenção, é possível reduzir significativamente o risco de vulnerabilidades de segurança em aplicações *web* e proteger os dados confidenciais dos utilizadores.

3.2.9 *Vulnerable and Outdated Components*

Um ataque de *Vulnerable and Outdated Components* em aplicações *web* ocorre quando um invasor aproveita as vulnerabilidades de componentes desatualizados ou com falhas de segurança numa aplicação para ganhar acesso indevido ou controlar a mesma. Esses componentes podem ser bibliotecas, *frameworks* ou *plugins* utilizados na construção da aplicação. Para prevenir esse tipo de ataque, é importante manter os componentes atualizados.

Entre as informações compartilhadas com o OWASP, cerca de 51,78% das aplicações *web* testadas apresentaram algum tipo de ataque direcionado a componentes desatualizados e vulneráveis. Esses ataques tiveram uma incidência máxima de 27,96%, média de 8,77% e totalizaram 30 mil ocorrências. É importante destacar que esta vulnerabilidade está relacionada a apenas 3 das vulnerabilidades do CWE.

3.2.9.1 *Vulnerabilidades*

Um *website* pode estar vulnerável a *Vulnerable and Outdated Components* quando:

- Não conhece as versões de todos os componentes utilizados, tanto do lado do cliente quanto do servidor, incluindo dependências em camadas.
- Utiliza *softwares* vulneráveis, sem suporte ou desatualizados, como sistemas operativos, servidores *web*, *Database Management Dystem* (DBMS), aplicações, API e todos os componentes, ambientes de tempo de execução e bibliotecas.
- Programadores de *software* não testam a compatibilidade de bibliotecas, atualizadas ou corrigidas.
- Não proteger as configurações dos componentes (consulte 3.2.8).

3.2.9.2 *Possíveis consequências*

A exploração de uma vulnerabilidade num componente desatualizado ou vulnerável pode permitir que um invasor assuma o controlo da aplicação, tendo acesso a dados

e funcionalidades que deveriam ser restritas. Isso pode levar a roubo de informações sensíveis, como dados de clientes ou informações de pagamento.

Além disso, a exploração de vulnerabilidades em componentes pode permitir a instalação de *malware* ou *backdoors* na aplicação, dando ao invasor acesso contínuo e persistente a sistemas e dados corporativos. Essas infeções podem ser utilizadsa para comprometer outras áreas da rede e para lançar ataques em outros sistemas ou aplicações.

Outra consequência comum de vulnerabilidades em componentes é a interrupção do serviço ou *Denial of Service* (DoS). Se um invasor explorar uma vulnerabilidade num componente crítico, ele pode fazer com que a aplicação inteira falhe ou fique inacessível, causando interrupções no negócio e perda de receita.

Por fim, a exploração de vulnerabilidades em componentes pode ter implicações legais e regulatórias. As organizações podem ser responsabilizadas por não proteger adequadamente os dados dos seus clientes e parceiros, e as violações de privacidade podem resultar em ações judiciais ou penalidades regulatórias.

3.2.9.3 *Como prevenir*

Para prevenir ataques por meio de *Vulnerable and Outdated Components*, é importante adotar medidas de segurança que reduzam as hipóteses de que uma vulnerabilidade seja explorada. Algumas medidas para prevenir esses tipos de ataques:

- Implementar um processo de gestão de *patches* para remover dependências não utilizadas, recursos desnecessários, componentes, ficheiros e documentação.
- Realizar inventário contínuo das versões dos componentes, tanto do lado do cliente quanto do lado do servidor (por exemplo, *frameworks* e bibliotecas) e as suas dependências. Monitorizar continuamente fontes como o *Common Vulnerability and Exposures* (CVE) e o *National Vulnerability Database* (NVD) na procura por vulnerabilidades nos componentes.
- Utilizar ferramentas de análise de composição de *software* para automatizar o processo.
- Inscrever-se em alertas para vulnerabilidades de segurança relacionadas aos componentes utilizados.
- Obter componentes apenas de fontes oficiais e por meio de *links* seguros. Preferir pacotes assinados para reduzir a hipótese de incluir um componente malicioso ou modificado (ver 3.2.11).

- Monitorizar bibliotecas e componentes que não são mantidos ou não criam *patches* de segurança para versões mais antigas. Se não for possível aplicar um *patch*, considerar a implantação de um *patch* virtual para monitorizar, detetar ou proteger contra o problema descoberto.

Cada organização deve garantir um plano contínuo para monitorizar, triar e aplicar atualizações ou alterações de configuração durante toda a vida útil da aplicação, ou portfólio.

3.2.10 *Identification and Authentication Failures*

A exploração de uma vulnerabilidade do tipo *Identification and Authentication Failures* em aplicações *web* é um tipo de ataque cibernético em que um invasor tenta explorar vulnerabilidades no sistema de identificação e autenticação de uma aplicação para ganhar acesso não autorizado a informações ou recursos protegidos. Isso pode ser feito por meio de técnicas como força bruta, ataques de senha, ataques de sessão ou falsificação de identidade.

Dos dados fornecidos ao OWASP, foi constatado que aproximadamente 79,51% das aplicações *web* testadas foram alvo de ataques voltados para componentes obsoletos e vulneráveis. Tais ataques apresentaram uma frequência máxima de 14,84%, uma média de 2,55% e totalizaram 132 mil ocorrências. É válido ressaltar que essa vulnerabilidade está associada a 22 tipos de vulnerabilidades identificadas pela CWE.

3.2.10.1 *Vulnerabilidades*

Confirmação da identidade do utilizador, autenticação e gestão de sessão são críticos para proteger contra ataques relacionados à autenticação. Para evitar fragilidades na autenticação, é necessário verificar os seguintes pontos:

- Não permitir ataques automatizados como preenchimento de credenciais ou ataques de força bruta.
- Não permitir senhas padrão, fracas ou bem conhecidas.
- Não utilizar processos de recuperação de credenciais e de senha esquecida que sejam fracos ou ineficazes.
- Não armazenar senhas em texto simples ou com *hashes* fracos.
- Utilizar segundo fator de autenticação de forma eficaz.
- Não expor o identificador de sessão na URL.

- Não reutilizar o identificador de sessão após o *login* bem-sucedido.
- Invalidar corretamente os IDs de sessão durante o *logout* ou um período de inatividade.
- Não comprometer a privacidade do utilizador ou a segurança dos dados durante a autenticação e gestão de sessão.

3.2.10.2 *Possíveis consequências*

Um ataque de *Identification and Authentication Failures* num *website* pode ter consequências graves para a segurança e a privacidade dos utilizadores. Os atacantes podem se aproveitar de vulnerabilidades no processo de autenticação para aceder informações confidenciais, realizar ações não autorizadas ou assumir o controlo da conta do utilizador.

Entre as consequências mais comuns de um ataque de *Identification and Authentication Failures* estão o roubo de informações de *login*, o acesso não autorizado a recursos protegidos, a manipulação ou destruição de dados do utilizador, a propagação de malware e a violação da privacidade do utilizador. Além disso, tais ataques podem prejudicar a reputação da empresa ou da aplicação *web* em questão, gerando perda de confiança dos utilizadores e impactando negativamente os negócios. Por isso, é essencial que as organizações adotem boas práticas de segurança para evitar vulnerabilidades relacionadas à autenticação e gestão de sessões.

3.2.10.3 *Como prevenir*

Como prevenir ataques de Identification and Authentication Failures:

- Implementar um segundo fator de autenticação sempre que possível para evitar ataques automatizados como preenchimento de credenciais, ataques de força bruta e reutilização de credenciais roubadas.
- Não utilize credenciais padrão, especialmente para utilizadores administrativos.
- Implementar verificação de senhas fracas, como testar novas senhas ou senhas alteradas relativamente à lista das 10.000 piores senhas.
- Garantir que o registo, a recuperação de credenciais e as rotas de API sejam protegidos contra ataques de enumeração de contas, utilizando as mesmas mensagens para todos os resultados.
- Limitar ou atrasar cada vez mais as tentativas de *login* falhadas, mas tenha cuidado para não criar um cenário de negação de serviço. Registar todas as falhas e alertar os administradores quando forem detetados preenchimento de credenciais, ataques de força bruta ou outros ataques.

- Utilizar um gestor de sessões incorporado e seguro no lado do servidor que crie um ID de sessão aleatório com alta entropia após o *login*. O identificador de sessão não deve estar no URL, ser armazenado com segurança e ser invalidado após o *logout*, período de inatividade e *timeouts* absolutos.

3.2.11 *Software and Data Integrity Failures*

Falhas de integridade de *software* e dados em aplicações *web*, ocorrem quando há erros ou corrupções nos dados ou no código do *software* que fazem com que a aplicação se comporte de forma inesperada ou indesejada. Algumas das causas comuns de falhas de integridade incluem erros de programação, falhas na validação de dados, vulnerabilidades de segurança e problemas de configuração do servidor. É importante que as empresas implementem medidas preventivas e de deteção para garantir a integridade do *software* e dos dados nas suas aplicações *web*.

Após o OWASP analisar os dados que lhe foram submetidos, foi constatado que 75,04% dos *websites* avaliados apresentaram alguma forma de ataque voltado para *Software and Data Integrity Failures*, com a maior incidência alcançando 16,67%, uma média de 2,05% e um total de quase 48 mil ocorrências. Essa vulnerabilidade engloba um total de 10 vulnerabilidades do CWE.

3.2.11.1 *Vulnerabilidades*

Falhas de integridade de *software* e dados se referem a códigos e infraestruturas que não protegem contra violações de integridade. Um exemplo disso é quando uma aplicação depende de *plugins*, bibliotecas ou módulos de fontes não confiáveis, repositórios e *Content Delivery Network* (CDN). Um pipeline CI/CD inseguro pode introduzir o potencial de acesso não autorizado, código malicioso ou comprometimento do sistema.

Muitas aplicações incluem funcionalidades de atualização automática, onde as atualizações são baixadas sem verificação suficiente de integridade e aplicadas à aplicação anteriormente confiável. Os atacantes podem potencialmente enviar as suas próprias atualizações para serem distribuídas e executadas em todas as instalações. Outro exemplo é quando objetos ou dados são codificados, ou serializados numa estrutura que um atacante pode ver e modificar, o que torna vulnerável à deserialização insegura.

3.2.11.2 *Possíveis consequências*

As consequências de um ataque derivado de falhas de integridade de software e dados podem ser graves e variadas. Alguns exemplos incluem:

- Roubo de informações confidenciais: Os atacantes podem explorar as vulnerabilidades para aceder às informações confidenciais, como dados pessoais, informações financeiras, senhas e informações de propriedade intelectual.
- Corrupção de dados: Os atacantes podem manipular ou excluir dados importantes, tornando-os inutilizáveis para a empresa ou os utilizadores.
- Interrupção de serviços: As falhas de integridade podem levar a interrupções de serviços, como *websites* ou aplicações que ficam indisponíveis para os utilizadores.
- Prejuízo financeiro: As falhas podem levar a custos de reparo ou recuperação de dados, além de perdas financeiras devido a violações de segurança, interrupções de serviços e perda de negócios.
- Dano à reputação: Os incidentes de segurança podem prejudicar a imagem da empresa e a confiança dos clientes na marca.
- Sanções legais: As empresas podem enfrentar sanções legais por não proteger adequadamente os dados dos utilizadores ou por violar as leis de proteção de dados.

3.2.11.3 *Como prevenir*

Existem várias medidas para conter possíveis ataques devido a vulnerabilidades do tipo de *Software and Data Integrity Failures*:

- Utilizar assinaturas digitais ou mecanismos semelhantes para verificar se o *software* ou os dados são provenientes da fonte esperada e não foram alterados.
- Certificar de que as bibliotecas e dependências, como *Node Package Manager* (NPM) ou *Maven*, estejam a consumir repositórios confiáveis. Se existir um perfil de risco mais alto, deve ser considerado hospedar um repositório interno de boa procedência que seja avaliado.
- Garantir que uma ferramenta de segurança de cadeia de suprimentos de *software*, como OWASP Dependency Check ou OWASP CycloneDX, seja utilizada para verificar se os componentes não contêm vulnerabilidades conhecidas.
- Garantir que exista um processo de revisão de alterações de código e configuração para minimizar a hipótese de que código ou configuração maliciosos possam ser introduzidos no seu *pipeline* de *software*.

- Garantir que o seu pipeline de CI/CD tenha uma segregação adequada, configuração e controlo de acesso para garantir a integridade do código que flui pelos processos de construção e implantação.
- Certificar de que dados serializados sem assinatura ou criptografia não sejam enviados para clientes não confiáveis sem algum tipo de verificação de integridade ou assinatura digital para detetar adulteração ou reprodução dos dados serializados.

3.2.12 *Security Logging and Monitoring Failures*

Security logging and monitoring failures em aplicações *web* referem-se a situações em que os sistemas de segurança de uma aplicação *web* falham em registar e monitorizar adequadamente atividades maliciosas ou suspeitas. Isso pode ocorrer devido a falhas técnicas, configurações inadequadas ou falta de recursos para lidar com o volume de dados gerados. Como resultado, as atividades maliciosas podem passar despercebidas pelos sistemas de segurança, permitindo que os invasores continuem a operar na aplicação sem serem detetados. A falta de registo e monitorização também dificulta a investigação e a resposta a incidentes de segurança, dificultando a identificação a origem e a extensão de um ataque e tomar medidas para prevenir futuros incidentes.

Conforme os dados fornecidos pelo OWASP, cerca de 53,67% das aplicações *web* testadas foram vítimas de ataques que exploram vulnerabilidades de *Identification and Authentication Failures*. Esses ataques apresentaram uma frequência máxima de 19,23%, uma média de 6,51% e somaram um total de 53 mil ocorrências. É importante destacar que essa vulnerabilidade está relacionada a 242 tipos diferentes de vulnerabilidades identificadas pela CWE.

3.2.12.1 *Vulnerabilidades*

Esta categoria do OWASP visa ajudar a detetar, aumentar a gravidade e responder a invasões ativas. Sem o registo e sem monitorização, as invasões não podem ser detetadas. A falta de registo, deteção, monitorização e resposta ocorrem sempre que:

- Eventos aditáveis, como *logins*, *logins* falhos e transações de valores, não são registados.
- Avisos e erros geram mensagens de registo insuficientes ou pouco claras.
- Os registos das aplicação e API não são monitorizados em procuras de atividades suspeitas.
- Os registos são armazenados apenas localmente.

- Não há limiares de alerta apropriados e processos de escalonamento de resposta não estão em vigor ou são ineficazes.

Ao tornar os eventos de registos e alertas visíveis para um utilizador ou atacante, fica vulnerável a fugas de informações (consultar 3.2.4).

3.2.12.2 *Possíveis consequências*

Explorar uma vulnerabilidade de *Security Logging and Monitoring Failures* pode ter consequências graves para qualquer aplicação. Isso ocorre porque o impedimento das aplicações ao registar atividades maliciosas ou suspeitas, permite que os invasores operem livremente sem serem detetados. Além disso, a falta de registo e monitorização dificulta a investigação e a resposta a incidentes de segurança, dificultando a identificar a origem e a extensão de um ataque e tomar medidas para prevenir futuros incidentes.

As possíveis consequências de explorar com sucesso uma vulnerabilidade de *Security Logging and Monitoring Failures* incluem perda ou roubo de dados sensíveis, comprometimento da integridade e disponibilidade da aplicação *web*, violação de privacidade do utilizador e danos à reputação da empresa. Além disso, os custos financeiros associados à recuperação e reparo de uma violação de segurança podem ser significativos. Por essas razões, é fundamental que as empresas implementem medidas adequadas de registo e monitorização de segurança nas suas aplicações *web*.

3.2.12.3 *Como prevenir*

Para prevenir este tipo de vulnerabilidade os programadores devem implementar os seguintes controlos de segurança, dependendo do risco de aplicação:

- Garantir que todas as falhas de *login*, controlo de acesso e validação de entrada do lado do servidor possam ser registadas com contexto de utilizador suficiente para identificar contas suspeitas ou maliciosas e mantidas por tempo suficiente para permitir análise forense posterior.
- Certificar de que os *logs* sejam gerados num formato que soluções de *Security Information and Event Management* (SIEM) possam consumir facilmente.
- Garantir que os dados dos *logs* sejam codificados corretamente para evitar injeções ou ataques nos sistemas de registo, ou monitorização.
- Estabelecer monitorização e alerta efetivos para detetar e responder rapidamente a atividades suspeitas.

- Estabelecer ou adotar um plano de resposta a incidentes e recuperação, como o NIST 800-61r2 ou posterior.
- Existem *frameworks* comerciais e de proteção de aplicações de código aberto, como o OWASP ModSecurity Core Rule Set[77], e SIEM de código aberto, como o Elasticsearch[78] com os módulos Logstash[79] e Kibana[80], que possuem painéis personalizados e alertas.

3.2.13 *Server Side Request Forgery*

SSRF é uma vulnerabilidade em aplicações *web* que permite que um atacante envie requisições maliciosas a partir do servidor *web* para outros sistemas internos ou externos, explorando uma falha na validação de entrada de dados. A exploração bem-sucedida de uma SSRF pode levar a graves consequências, tais como o comprometimento completo de um sistema ou a perda de dados críticos.

Segundo os dados divulgados pelo OWASP, aproximadamente 67,72% das aplicações *web* testadas foram alvo de ataques que exploram vulnerabilidades de SSRF. Esses ataques ocorreram com uma frequência máxima de 2,72%, uma média de 2,72%, totalizando 9.503 ocorrências. É importante ressaltar que essa vulnerabilidade está relacionada a apenas uma das vulnerabilidades identificadas pela CWE.

3.2.13.1 *Vulnerabilidades*

As vulnerabilidades de SSRF ocorrem sempre que uma aplicação *web* procura um recurso remoto sem validar o URL fornecido pelo utilizador. Isso permite que um atacante force a aplicação a enviar uma requisição manipulada para um destino inesperado, mesmo quando protegido por uma *firewall*, *Virtual Private Network* (VPN) ou outro tipo de ACLs de rede.

Como as aplicações *web* modernas fornecem recursos convenientes aos utilizadores finais. Como resultado, a incidência de SSRF aumenta. Além disso, a gravidade do SSRF torna-se maior devido aos serviços em nuvem e à complexidade das arquiteturas de rede.

3.2.13.2 *Possíveis consequências*

É importante destacar que a vulnerabilidade de SSRF pode ser utilizada como parte de um ataque mais amplo, combinada com outras vulnerabilidades, para aumentar o impacto e a severidade do ataque. Algumas das possíveis consequências incluem:

- Acesso não autorizado a informações confidenciais, como credenciais de utilizador, chaves de API, *tokens* de autenticação, entre outros;
- Exploração de serviços internos da aplicação ou da rede, como bases de dados, sistemas de ficheiros, sistemas de gestão de filas, entre outros;
- Execução remota de código malicioso no servidor ou em outros sistemas da rede;
- Ataques a outros sistemas externos a partir do servidor comprometido;
- Modificação ou exclusão de dados importantes e sensíveis;
- Comprometimento total do sistema e perda de controlo sobre os dados armazenados.

3.2.13.3 *Como prevenir*

Os programadores podem prevenir a vulnerabilidade de SSRF implementando os seguintes controlos de defesa em profundidade:

Camada de Rede

- Segmentar a funcionalidade de acesso a recursos remotos em redes separadas para reduzir o impacto do SSRF .
- Aplicar políticas de *firewall* "*deny by default*" ou regras de controlo de acesso de rede para bloquear o tráfego *intranet*, exceto o essencial.

Camada de aplicação

- Validar todos os dados de entrada fornecidos pelo cliente.
- Aplicar *white lists* para esquema de URL, porta e destino.
- Não enviar respostas cruas aos clientes.
- Desativar redirecionamentos HTTP.
- Estar ciente da consistência da URL para evitar ataques como o *Domain Name System* (DNS) *rebinding* e as condições de corrida "*time of check, time of use*".

3.2.14 *Cross-Site Request Forgery*

O Vue.js, assim como outras *frameworks frontend*, possui suporte nativo para proteção contra ataques CSRF. Ao utilizar o *Axios*, por exemplo, este já inclui automaticamente o *token* CSRF nas requisições HTTP enviadas para o *backend*. Isso

é feito utilizando o *cookie* CSRF *token*, criado pelo Laravel durante a autenticação do utilizador e enviado ao navegador como uma resposta para o cliente.

O *Axios*, por padrão, adiciona automaticamente o *token* CSRF como um cabeçalho HTTP designado "*X-CSRF-TOKEN*"em todas as requisições *Create, Read, Update and Delete* (CRUD). O Laravel verifica automaticamente a presença deste cabeçalho na requisição e compara com o valor do *cookie* CSRF *token* para verificar se a requisição é válida. Se os valores coincidirem, a requisição é processada normalmente. Caso contrário, o Laravel retornará um erro 419 - *CSRF Token Mismatch*.

3.3 PENTEST

O processo de PenTest é uma prática crucial na área de segurança cibernética, projetada para avaliar a segurança de sistemas de informação, simulando um ataque de um utilizador mal-intencionado.

Ao realizar o PenTest, as organizações podem identificar vulnerabilidades e pontos fracos nos seus sistemas, redes ou aplicações, possibilitando a implementação de medidas corretivas antes que essas falhas sejam exploradas por agentes mal-intencionados. No entanto, para garantir a eficácia e a abrangência do processo de teste, é imperativo seguir uma metodologia estruturada.

Embora existam várias metodologias reconhecidas na temática de cibersegurança, conforme abordado na secção 2.3, adaptar ou combinar abordagens específicas pode oferecer uma visão mais holística e aprofundada das vulnerabilidades presentes, possibilitando uma análise mais completa e precisa do ambiente de segurança.

Ao considerar a complexa paisagem da segurança cibernética, torna-se evidente que a seleção de uma metodologia específica para testes de penetração não pode ser uma abordagem de tamanho único. Cada ambiente e sistema apresentam desafios únicos e requerem uma compreensão profunda das particularidades da infraestrutura e das aplicações envolvidas. Uma metodologia bem escolhida é crucial para garantir a eficácia e a abrangência dos testes de penetração.

3.3.1 *Estudo das metodologias de PenTest*

A análise realizada nesta secção é entre as metodologias mencionadas na secção *metodologias de penetration testing* 2.3. A seguir são apresentados os pontos principais de cada metodologia para o cenário de PenTest na aplicação *web*.

A diversidade de metodologias oferece um conjunto diversificado de diretrizes, técnicas e práticas recomendadas. Cada uma dessas metodologias aborda aspetos

específicos dos testes de penetração, desde a segurança física e humana até as vulnerabilidades específicas de aplicações *web*.

A compreensão profunda e a análise cuidadosa das características individuais de cada metodologia permitem extrair elementos cruciais que podem ser relacionados para criar uma abordagem personalizada e adaptada. A fusão estratégica de diversos métodos pode não apenas aprimorar a precisão e a abrangência dos testes de penetração, mas também promover uma compreensão mais holística dos riscos de segurança, permitindo a implementação de medidas proativas e direcionadas para fortalecer a postura de segurança geral.

No contexto de metodologias de testes de penetração, várias abordagens merecem consideração. A metodologia OSSTMM destaca-se pela sua abordagem holística, que abrange não apenas testes técnicos, mas também avaliações de segurança física e humana. No entanto, ao considerar uma aplicação *web* específica, é crucial notar que a OSSTMM pode necessitar de ajustes para forcar-se nas vulnerabilidades, dada a sua abrangência original.

Por outro lado, a OWASP WSTG concentra-se especificamente em testes de segurança para aplicações *web*, oferecendo uma ampla gama de técnicas e abordagens para identificar vulnerabilidades comuns nesse contexto. Enquanto isso, o NIST SP 800–115 oferece uma abordagem sistemática e detalhada para testes de penetração, devido ao seu foco principal não seja em aplicações *web*, poderá necessitar de adaptações para atender aos requisitos de segurança exclusivos desse ambiente.

A metodologia PTES, reconhecida por sua abordagem estruturada e completa, fornece diretrizes detalhadas para a execução de testes de penetração em vários ambientes, sendo facilmente adaptável para atender aos requisitos específicos de segurança de aplicações *web*.

A ISSA oferece uma variedade de recursos e diretrizes abrangentes para profissionais de segurança da informação, embora não seja especificamente uma metodologia de teste de penetração, as suas diretrizes podem informar práticas recomendadas e posturas gerais de segurança durante todo o processo de teste.

3.3.2 *Possíveis abordagens de PenTest*

Ao escolher uma metodologia de PenTest é necessário perceber qual o nível de visibilidade no cenário. Os PenTest por norma podem ter 3 tipos diferentes de visibilidade:

- *Black Box*: Nesse tipo de teste de penetração, os *testers* têm um conhecimento limitado ou nenhum conhecimento prévio do sistema. O sistema é abordado

por um atacante externo sem informações privilegiadas. Isso ajuda a simular um ataque realista de um invasor não autorizado. Os *testers* precisam realizar uma análise completa do sistema, identificando vulnerabilidades sem ter acesso ao código-fonte ou à arquitetura interna do sistema.

- *Grey Box*: O teste de caixa cinzenta é uma abordagem que combina elementos do teste de caixa preta e de caixa branca. Os *testers* têm um conhecimento parcial do sistema, mas não possuem um entendimento completo da infraestrutura de TI ou do código-fonte. Eles têm acesso limitado a informações sobre o sistema, simulando uma situação em que o atacante possui um conhecimento parcial da infraestrutura do sistema alvo. Essa abordagem equilibra os aspetos realistas do teste de caixa preta com a profundidade do teste de caixa branca.
- *White Box*: Neste caso, os *testers* têm conhecimento completo da infraestrutura de TI, do código-fonte e da arquitetura do sistema. Estes utilizam informações privilegiadas para aprofundar a análise e identificar possíveis pontos de vulnerabilidade. O teste de caixa branca permite uma avaliação mais detalhada e específica das vulnerabilidades internas do sistema, fornecendo uma visão mais precisa de possíveis riscos de segurança.

Ao realizar um teste de penetração numa aplicação *web* desenvolvida internamente, a escolha de uma abordagem de teste de caixa cinza (*grey box*) pode oferecer uma série de benefícios distintos. A familiaridade com a arquitetura, o código-fonte e a infraestrutura subjacente da aplicação *web* proporciona uma vantagem significativa ao permitir um entendimento aprofundado do contexto operacional e do *design* de segurança implementado.

Essa visão interna pode facilitar a identificação de possíveis pontos de vulnerabilidade e a compreensão das áreas críticas que requerem uma atenção especial durante o teste. Ao mesmo tempo, a aplicação da abordagem de teste de caixa cinza simula uma situação realista em que um atacante externo possui um conhecimento parcial do sistema alvo. Isso permite que os *testers* avaliem a resistência da aplicação *web* a ameaças comuns, considerando, possíveis lacunas de segurança que poderiam ser exploradas por atacantes mal-intencionados.

Além disso, a abordagem de teste de caixa cinza equilibra a profundidade do teste de caixa branca com a imparcialidade e a simulação realista de um teste de caixa preta, permitindo uma avaliação abrangente e precisa da segurança da aplicação *web*. A seleção do teste de caixa cinza, nesse contexto, pode oferecer uma perspetiva crítica e informada para fortalecer a postura de segurança da aplicação *web*, identificando e abordando proativamente possíveis riscos e vulnerabilidades.

3.3.3 *Metodologias escolhidas*

A decisão de combinar as metodologias PTES e OWASP WSTG para conduzir um teste de penetração numa aplicação *web*, surge da necessidade de alcançar uma avaliação abrangente e especializada da segurança da aplicação *web*. A escolha da metodologia PTES é fundamentada na sua abordagem estruturada e abrangente, que fornece diretrizes detalhadas para a execução de testes de penetração numa variedade de ambientes e sistemas.

A estrutura bem definida da PTES estabelece um fluxo de trabalho claro e organizado, desde a fase de planeamento até a execução e documentação, contribuindo para a eficácia e a coerência do processo de teste. Além disso, a PTES permite uma abordagem adaptável, facilitando a personalização das diretrizes para atender aos requisitos específicos de segurança de uma aplicação *web* em particular.

Por outro lado, a inclusão da metodologia OWASP WSTG é motivada pela sua especialização em testes de segurança de aplicações *web*. A coleção de técnicas e *checklists* fornecidas pela WSTG oferece um conjunto abrangente de diretrizes específicas para identificar vulnerabilidades comuns em aplicações *web*, como injeção SQL, XSS, entre outras.

A orientação detalhada e especializada fornecida pela *checklist* do OWASP WSTG permite uma abordagem minuciosa e sistemática ao avaliar a segurança de uma aplicação *web*, garantindo que as vulnerabilidades sejam identificadas e abordadas de forma precisa e abrangente.

A fusão estratégica dessas duas metodologias não apenas aprimora a profundidade e a amplitude do teste de penetração, mas também promove uma compreensão mais completa das ameaças e vulnerabilidades específicas que uma aplicação *web* pode enfrentar. A sinergia resultante dessas abordagens combinadas oferece uma estrutura robusta e adaptável para a execução de um teste de penetração eficaz e abrangente, permitindo a identificação proativa e a mitigação de possíveis riscos de segurança, fortalecendo assim a postura geral de segurança da aplicação *web* em análise.

3.3.4 *Estrutura PTES*

O PTES segue 7 etapas distintas para garantir uma avaliação abrangente da segurança da aplicação. A estrutura de realização dos *pentests* está representada na Figura 14.

Figura 14: Estrutura do PTES [81]

A seguir, é apresentada sucintamente cada uma das etapas aplicáveis a qualquer sistema para um processo eficaz de PenTest, seguindo o PTES.

3.3.4.1 *Pre-engagement Interactions*

A etapa de interações prévias ao compromisso no PTES visa estabelecer uma base sólida para o PenTest, permitindo uma compreensão clara das metas e objetivos do cliente. Esta fase não deve ser um confronto, mas sim uma colaboração para identificar os riscos associados a possíveis ataques.

Durante o estágio inicial, a equipa de teste e o cliente envolvem-se em discussões detalhadas para definir o âmbito do teste, considerando as limitações e restrições acordadas mutuamente. A interação visa garantir que todas as questões pertinentes sejam abordadas para alcançar o máximo valor e compreensão do contexto específico da organização em questão.

3.3.4.2 *Intelligence Gathering*

A fase de recolha de informações no PTES abrange atividades essenciais de inteligência. Esta etapa é fundamental para os *testers* de PenTest que realizam uma análise minuciosa do alvo, seja uma organização corporativa, militar ou similar. O documento define o processo de pensamento e os objetivos do reconhecimento do PenTest, com o propósito de ajudar a desenvolver um plano estratégico altamente eficaz para atacar um alvo específico.

Durante essa etapa, a equipa de teste dedica-se à recolha de uma ampla gama de informações. Isso pode incluir a procura de dados de fontes de domínio público, análise de registos *WHOIS*, investigação de perfis de redes sociais, recolha de informações sobre a empresa em questão e exploração de configurações de rede disponíveis publicamente.

O objetivo é reunir informações cruciais que possam oferecer uma compreensão aprofundada da infraestrutura do sistema de destino, proporcionando uma base sólida para as fases subsequentes do PenTest.

3.3.4.3 *Threat Modeling*

A etapa de modelagem de ameaças no processo de PenTest envolve uma abordagem rigorosa e abrangente na análise e identificação de potenciais ameaças que impactem o sistema ou a rede em questão. Durante esta fase, a equipa de teste avalia a estrutura de possíveis cenários de ameaças específicos, tendo como base as vulnerabilidades potenciais previamente descobertas durante a fase de coleta de informações.

Esta etapa desempenha um papel crucial ao direcionar os esforços da equipa de teste para áreas específicas que possuem maior suscetibilidade a riscos. O foco é não apenas nas vulnerabilidades técnicas, mas também na compreensão abrangente das capacidades e motivações dos potenciais agentes de ameaças, com a avaliação de ativos críticos e processos de negócios relevantes para a organização.

Este processo de modelação de ameaças proporciona uma base sólida para a conceção de estratégias de teste de penetração que visam replicar de forma eficaz as táticas, técnicas e objetivos dos potenciais atacantes. Garantindo uma abordagem focalizada e precisa durante as fases subsequentes do processo de avaliação.

3.3.4.4 *Vulnerability Analysis*

A etapa de análise de vulnerabilidades no processo de PenTest incluí uma análise mais ampla e sistemática para identificar e avaliar potenciais ameaças que comprometam a segurança do sistema ou rede em análise. Durante esta fase crítica, a equipa de teste procura ativamente por falhas de segurança, vulnerabilidades e fragilidades que possam ser exploradas por possíveis atacantes.

A avaliação de vulnerabilidades engloba uma série de técnicas, como análise de portas, análise de código, testes de penetração de rede e outros métodos específicos destinados a identificar e compreender potenciais pontos fracos. É fundamental para a equipa de teste delimitar adequadamente o âmbito do teste, considerando a profundidade e a amplitude necessárias para atender aos objetivos e requisitos de resultado desejados. A profundidade de teste envolve considerações sobre a

localização de ferramentas de avaliação, requisitos de autenticação e validação da eficácia das medidas de mitigação existentes.

Além disso, a amplitude da avaliação de vulnerabilidades abrange a extensão da análise, incluindo redes de destino, segmentos, hospedeiros, aplicações e inventários relevantes. Garantir a validação da amplitude do teste é essencial para garantir que o escopo do teste tenha sido completamente abordado e que todas as máquinas ou componentes pertinentes tenham sido incluídos na avaliação. Essa abordagem meticulosa ajuda a identificar uma gama mais ampla de possíveis ameaças, permitindo que a equipa de teste direcione os seus esforços para corrigir e fortalecer as áreas mais suscetíveis a riscos de segurança.

3.3.4.5 *Exploitation*

A fase de exploração num PenTest concentra-se exclusivamente em estabelecer o acesso a um sistema ou recurso, contornando as restrições de segurança. Se a fase anterior de análise de vulnerabilidades foi realizada adequadamente, esta etapa deve ser cuidadosamente planeada e executada com precisão. O foco principal é identificar o ponto principal de entrada na organização e os ativos de alto valor.

Durante esta etapa crítica, a equipa de teste visa explorar ativamente as vulnerabilidades identificadas de modo a obter acesso não autorizado ao sistema ou aos dados. Este passo é fundamental para avaliar o quão facilmente um atacante mal-intencionado pode aproveitar as falhas descobertas durante a fase de análise de vulnerabilidades. No entanto, é essencial respeitar estritamente o âmbito previamente definido para o PenTest, garantindo que as atividades de exploração sejam conduzidas de forma ética e dentro dos limites acordados na primeira etapa do PenTest.

3.3.4.6 *Post Exploitation*

A fase de pós-exploração aspira determinar o valor da máquina comprometida e manter o controlo sobre ela para uso posterior. O valor da máquina é avaliado com base na sensibilidade dos dados armazenados nela e na sua utilidade para comprometer ainda mais a rede.

Os métodos descritos nesta fase visam auxiliar os *testers* a identificar e documentar dados sensíveis, configurar definições de configuração, canais de comunicação e relacionamentos com outros dispositivos de rede que possam ser utilizados para obter acesso adicional à rede. Além disso, visa estabelecer um ou mais métodos de acesso à máquina em momentos posteriores. Em casos em que esses métodos diferem das regras acordadas, estas regras devem ser seguidas.

Após obter acesso ao sistema ou rede, a equipa de teste visa manter esse acesso e explorar mais informações sensíveis ou áreas adicionais que possam ser comprometidas. Esta fase é essencial para avaliar a extensão do dano que um invasor mal-intencionado pode causar se conseguir aceder e manter o controlo sobre o sistema. O objetivo é identificar quaisquer dados críticos que possam ser comprometidos, analisar as configurações de segurança existentes e estabelecer a persistência no sistema comprometido para manter o acesso a longo prazo. O foco também é garantir que as atividades realizadas durante esta fase estejam conforme as regras de ajuste acordadas, garantindo a ética e a legalidade do processo de PenTest.

3.3.4.7 *Reporting*

O processo de elaboração de relatórios em PenTest é crucial para comunicar eficazmente os resultados e recomendações aos clientes. Embora seja encorajado o uso de formatos personalizados e com marca própria, o relatório deve conter elementos essenciais para transmitir claramente as descobertas e agregar valor para o leitor.

Nesta fase final, a equipa de teste consolida todas as descobertas e resultados num relatório detalhado. O relatório aborda todas as vulnerabilidades identificadas, descreve os métodos utilizados durante o processo de teste, e oferece uma análise aprofundada dos riscos associados.

Além disso, o relatório pode incluir recomendações específicas para fortalecer a segurança do sistema, com medidas de mitigação detalhadas. O objetivo é fornecer ao cliente *insights* claros sobre as áreas de melhoria necessárias e estratégias práticas para reforçar a postura de segurança global da organização.

3.3.5 *OWASP WSTG*

A *checklist* OWASP WSTG é uma ferramenta essencial para realizar e acompanhar testes de segurança na *web*. No repositório do *github* da WSTG está disponível um Excel que contém diversas folhas que servem para facilitar o acompanhamento do progresso, a criação de tabelas de resultados e recomendações, bem como a avaliação de riscos. Este ficheiro é divido nas seguintes 4 folhas:

- *Testing Checklist* (Lista de Verificação de Testes): Esta folha de trabalho permite o acompanhamento do progresso relativamente a cada ponto do PenTest delineados no guia. Serve como um recurso valioso para garantir que cada etapa do PenTest seja concluída de forma abrangente e sistemática.
- *Summary Findings* (Resultados Resumidos): Esta folha de trabalho facilita a criação de uma tabela com os resultados dos PenTest e potenciais recomen-

dações. Ajuda a consolidar e resumir as descobertas-chave, oferecendo uma visão geral clara das áreas de preocupação e possíveis ações corretivas.

- *Risk Assessment Calculator* (Calculadora de Avaliação de Riscos): Esta é uma folha responsável por calcular os índices de probabilidade e impacto, além de uma classificação geral qualitativa de risco. Esta funcionalidade ajuda a avaliar a gravidade das potenciais vulnerabilidades identificadas durante o processo de PenTest.
- *References* (Referências): Esta folha fornece listas e conjuntos de dados nos quais o calculador de riscos se baseia, fornecendo um contexto para a avaliação e a classificação de riscos.

Esta *checklist* baseia-se na versão 4.2 do OWASP WSTG, uma vez que o conteúdo para a versão 5 continua em desenvolvimento. Ela serve como um recurso valioso para garantir a abrangência e a eficácia dos testes de segurança *web*, seguindo as melhores práticas recomendadas pelo OWASP.

3.4 SÍNTESE

Neste capitulo, foi detalhado o contexto do ITS para a implementação de uma aplicação *web* para gerir os sinais rodoviários. A segurança dos dados ITS na aplicação *web* é enfatizada, destacando os riscos. Foi realizada a análise às organizações de segurança escolhidas, OWASP *top* 10 e o CWE *top* 25. Foi analisado o procedimento para a realização do PenTest enquadrado neste projeto, utilizando a metodologia PTES e a *checklist* do OWASP WSTG.

No próximo capítulo (4), são detalhados os requisitos para o desenvolvimento deste projeto com o enquadramento do ITS. O capítulo ainda incluiu os mecanismos e funcionalidades implementadas para garantir a segurança da aplicação, seguindo as melhores práticas do OWASP.

4

DESENVOLVIMENTO

O capítulo de desenvolvimento, é estruturado em duas fases essenciais. Na primeira, são apresentadas as funcionalidades da aplicação, alinhadas aos princípios do ITS, visando uma gestão eficiente dos sinais rodoviários na aplicação *web*. A segunda fase destaca os mecanismos e funcionalidades de segurança incorporados para proteger a aplicação, seguindo as melhores práticas do OWASP. Estas duas fases constituem a base do desenvolvimento, integrando teoria e prática de forma concisa e eficiente.

4.1 FUNCIONALIDADES DA APLICAÇÃO

Os requisitos para o desenvolvimento desta plataforma vão ao encontro da ISO14823 sobre o ITS. Antes de iniciar a construção das funcionalidades da aplicação foi necessário estruturar os requisitos e perceber que tipos de dados seriam relevantes para a aplicação.

O objetivo principal do projeto foi o desenvolvimento de uma aplicação que suportasse a implementação de sinais de forma intuitiva e simples para o utilizador, criando dados ITS confiáveis para serem utilizados noutras aplicações.

Com este propósito foi criada uma base de dados estruturada para armazenar os dados aplicacionais e com capacidade para suportar a estrutura de mensagens IVIM. Em simultâneo, a base de dados também teve de suportar as funcionalidades da aplicação, relacionadas com os utilizadores e com as entidades.

4.1.1 *Histórico da aplicação*

A aplicação *web* não foi criada de raiz, existia um protótipo funcional, com uma estrutura simplificada que permitia criar os sinais rodoviários no mapa e permitia parte da gestão dos utilizadores. A aplicação final utilizou o *layout* desenvolvido no protótipo. Apesar de existirem funcionalidades muito idênticas, foi necessário reconstruí-la devido à criação da nova base de dados e aos mecanismos de segurança, tais como utilização de *tokens* nos pedidos da API para validar o utilizador e validações os *inputs* dos formulários da aplicação. Na Figura 15 está representado o mapa do protótipo da aplicação.

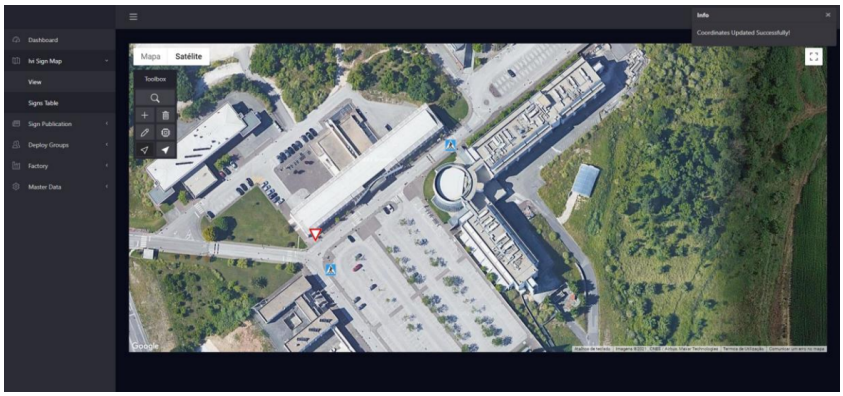


Figura 15: Protótipo da aplicação

A aplicação *web* foi desenvolvida com a *framework* Laravel para o *backend* e a *framework* Vue.js para o *frontend*. A utilização destas *frameworks* foi mantida, no entanto, foi necessário atualizar as *frameworks*. Vários pacotes estavam *deprecated* ou desatualizados, sendo necessário encontrar alternativas ou simplesmente atualizar os pacotes.

Inicialmente foi realizado um estudo para perceber quais as alterações necessárias na base de dados. As tabelas existentes foram quase todas removidas, permanecendo apenas tabelas relacionadas com o menu da aplicação. Todas as outras tabelas sofreram grandes alterações ou acabaram mesmo por serem removidas. A fim de avaliar a dimensão das duas bases de dados, a antiga continha 37 tabelas, enquanto que a nova possui 197 tabelas.

A API existente também sofreu alterações devido à segurança da aplicação, foi criado um mecanismo de segurança para permitir pedidos sem autenticação, como os pedidos relacionados ao *log in* e à recuperação de *password*. Outro mecanismo criado foi para os pedidos que requerem autenticação, mas sem a confirmação do utilizador, por exemplo, para enviar o código de verificação ou para confirmar o segundo fator de autenticação.

Além das modificações de segurança que a API sofreu, houve a necessidade de modificar diversas solicitações para interagir com outras ferramentas externas à aplicação e modificações necessárias, devido a mudanças nas funcionalidades da plataforma.

4.1.2 *Base de dados*

A base de dados sustenta a aplicação *web* e desempenha um papel fundamental no armazenamento e na organização dos dados gerados pela aplicação *web*. Esta base de dados, desenvolvida em MySQL, foi projetada para armazenar informações

relacionadas à estrutura de mensagem IVIM e dados essenciais para o funcionamento da aplicação.

Para lidar com a complexidade da base de dados, foi criado um ficheiro no MySQL WorkBench que permite ter uma interface gráfica da base de dados, designado como *model* pela aplicação e criado com a extensão *.mwb*. Este modelo foi elaborado para representar as tabelas e todas as suas relações visualmente para o programador. Sempre que foi necessário efetuar alguma alteração na estrutura da base de dados, essa modificação foi realizada no modelo por meio do MySQL Workbench.

A conexão com o MySQL é estabelecida via Docker na porta 3306, e a versão do MySQL utilizada é a 8.0. As informações de autenticação desta conexão, que inclui o *username* e *password*, foram configuradas no ficheiro *.env*, garantindo a segurança da conexão e dos dados armazenados.

A base de dados tem 197 tabelas interrelacionadas, cada tabela foi cuidadosamente projetada para representar os elementos específicos da estrutura da mensagem IVIM e em simultâneo suportar os requisitos da aplicação *web*. No Apêndice A, está representado o diagrama da base de dados.

Durante o processo de desenvolvimento, foram encontrados desafios relacionados à migração da base de dados antiga para a estrutura da aplicação *web* inicial. A necessidade de manter o funcionamento da aplicação, ao mesmo tempo, em que era incorporada a estrurua IVIM, levou à criação de novas tabelas e à eliminação das antigas, sendo muito desafiante. Este processo meticuloso garantiu que os dados antigos fossem mantidos apropriadamente e que os novos dados fossem integrados coerentemente.

A segurança dos dados foi uma prioridade durante o processo de desenvolvimento. Cada tabela é identifica por um campo “id” ou “id_x” para relações, onde “x” representa o nome da tabela relacionada. Além disso, a escolha dos tipos de dados para os diferentes campos, bem como a definição de tamanhos apropriados para os campos, foram considerações essenciais para manter a precisão dos dados armazenados.

A base de dados é o núcleo vital da aplicação. Todas as informações necessárias para o funcionamento da aplicação, bem como os dados relacionados à estrutura IVIM, são armazenados nela. Além disso, a base de dados é acedida por meio de controladores da *framework* Laravel, que comunicam através dos modelos para aceder e manipular os dados armazenados na base de dados. Isto garante que a aplicação tenha um acesso rápido e eficiente aos dados necessários para o seu correto funcionamento.

Atualmente, a aplicação está em fase de pré-produção, e a base de dados está pronta para suportar as operações da aplicação para a fase de produção. A solidez

da base de dados é um elemento essencial para o sucesso contínuo da aplicação. A complexidade da estrutura, os desafios enfrentados e as medidas de segurança implementadas destacam a importância crítica da base de dados no ecossistema da plataforma *web*.

4.1.3 *API*

A API desempenha um papel crucial na comunicação entre o *frontend*, desenvolvido em Vue.js, e o *backend* construído em Laravel. Além de atender às necessidades da aplicação *web*, a API também está preparada para responder a solicitações de outras aplicações externas.

A API segue um padrão de *design RESTful*, o que significa que adere aos princípios de representação de recursos e comunicação *stateless*. Essa arquitetura orientada a recursos simplifica a interação entre o cliente (*frontend*) e o servidor (*backend*) por meio de URL e métodos HTTP.

A segurança é uma preocupação fundamental no desenvolvimento da API. Foram implementas medidas de autenticação e autorização para proteger os dados e recursos sensíveis. Existem pedidos que são atendidos sem a necessidade de autenticação nem de autorização, como o *login*. Pedidos que necessitam de autenticação, mas que não necessitam de autorização, como os pedidos de confirmação de identidade do utilizador. E o terceiro cenário, que corresponde à maioria dos pedidos, que requerem tanto autenticação quanto autorização para garantir que apenas os utilizadores autorizados tenham acesso a recursos específicos.

A API não interage diretamente com a base de dados. Em vez disso, a aplicação *web* envia solicitações à API, que, por sua vez, executa controladores que acedem a modelos da *framework* Laravel para interagir com a base de dados. Esta abordagem garante uma camada de abstração entre a API e os dados subjacentes, facilitando a manutenção e a escalabilidade.

A utilização de *middlewares* permite controlar quais os pedidos que requerem autenticação e/ou autorização. Isso simplifica a adição de novos recursos à medida que a aplicação cresce, garantindo que a segurança e o desempenho não sejam comprometidos.

Embora a maioria dos *endpoints* da API atenda aos pedidos da aplicação *web*, alguns *endpoints* específicos foram criados para atender a aplicações externas. Essas integrações externas fornecem dados para a criação de ficheiros binários, contribuindo para a versatilidade da API. Alguns destes *endpoints* têm o intuito de serem utilizados pelos OBU.

A API desempenha um papel fundamental na arquitetura da aplicação, facilitando a comunicação entre os diferentes componentes do sistema. A sua natureza *RESTful*, foco na segurança e escalabilidade garantem que ela esteja preparada para atender às necessidades presentes e futuras da plataforma.

4.1.4 *Autenticação*

O método de autenticação requer as credenciais de acesso do utilizador, sendo estas o endereço de correio eletrónico e a palavra-passe. Qualquer utilizador que não esteja autenticado encontra a página de autenticação e apenas consegue aceder à página de recuperar a palavra-passe, sem conseguir aceder às restantes funcionalidades da aplicação. Na Figura 16 é possível ver a página de autenticação.

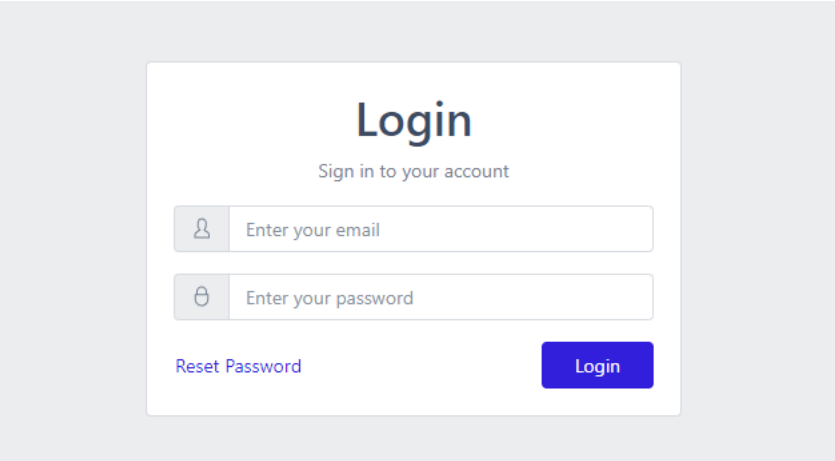


Figura 16: Página de autenticação

A autenticação exigiu a criação de vários mecanismos de segurança descritos na secção 4.2. Para a construção de alguns desses mecanismos de segurança foi necessário criar funcionalidades adicionais na autenticação do utilizador.

Posteriormente ao utilizador autenticar-se tem de confirmar a sua conta por meio de um segundo fator de autenticação, podendo ser via aplicação móvel ou por endereço de correio eletrónico.

Na primeira autenticação do utilizador a aplicação permite o utilizador configurar o *Multifactor Authentication* (MFA) via aplicação móvel ou optar por receber um *email* com um código sempre que o mesmo se autentica. Esta página é mostrada na Figura 17.

Quando não é a primeira vez que o utilizador se autentica, é apresentada a página de verificação de código por aplicação ou a página de verificação de código por endereço de correio eletrónico. O utilizador pode alterar esta configuração no perfil do utilizador.

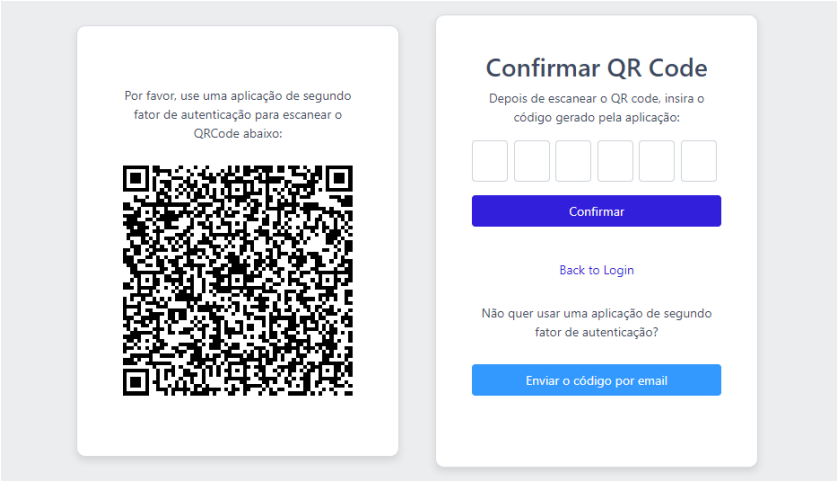


Figura 17: Página da primeira autenticação

4.1.5 *Recuperar palavra-passe*

Quando o utilizador não consegue realizar a autenticação, pode aceder à página de recuperação de palavra-passe. Inicialmente é redirecionado para uma página de confirmação de identidade, onde o utilizador terá de inserir o seu endereço de correio eletrónico e o número de telemóvel associado à sua conta da plataforma.

Esta página tem validações nos dados inseridos pelo utilizador e mecanismos de bloqueio para ataques de *brute force* que irão ser abordados na secção 4.2.

Após a inserção bem sucedida dos dados do utilizador, é enviado um *email* para o utilizador com um código de verificação e é redirecionado para a página de verificação.

Nesta página o utilizador deverá colocar o código recebido no endereço de correio eletrónico. Este código é válido apenas por 5 minutos. Assim que o código for validado, o utilizador é redirecionado para a última página do processo de recuperação de palavra-passe, apresentada na Figura 18.

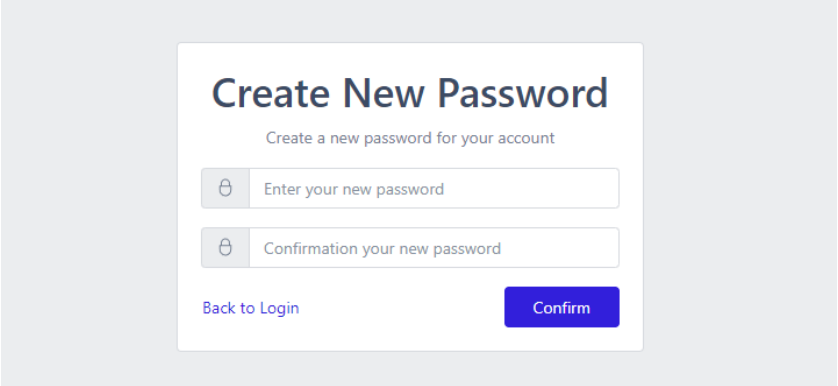


Figura 18: Reposição de palavra-passe

O utilizador deverá criar uma nova palavra-passe para a sua conta. Esta palavra-passe deverá conter pelo menos uma letra maiúscula e minúscula, um carácter especial e um número. E o seu tamanho deverá possuir no mínimo 12 carateres. Assim que o utilizador confirmar a nova palavra-passe, será redirecionado para a página de *log in*.

4.1.6 *Funcionalidades da aplicação*

Quando o utilizador se autêntica e confirma a sua identidade, tem acesso a diversas funcionalidades da aplicação que irão ser abordadas nas seguintes subsecções. Na Figura 19 é possível analisar o menu da aplicação que permite aceder às funcionalidades da aplicação.

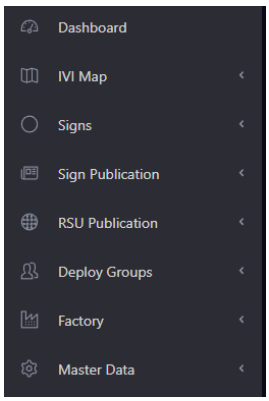


Figura 19: Menu da aplicação

Cada página apresentada neste menu será abordada, explicando as funcionalidades principais da plataforma. Este menu com estas opções é a visão de qualquer conta com privilégios de administrador. Conforme as permissões dos utilizadores, o menu não será tão composto, mostrando apenas as opções para que o utilizador tem permissões.

Este tópico de permissões de utilizadores será abordado na 4.2.1 porque faz parte da temática de segurança da aplicação.

4.1.7 *Dashboard*

A página "Dashboard" representa informações da plataforma em formato de gráficos. Reúne informações dispersas numa única página, proporcionando uma visão panorâmica que auxilia na compreensão dos dados do sistema.

Esta página é composta por 7 gráficos:

- Número total de sinais e de RSU;
- Número de sinais por cada categoria.
- Número de zonas configuradas nos sinais, deteção, relevância ou de conhecimento.
- Número de utilizadores por *roles*.
- Número de sinais por tipologia, *vienna* ou *iso*.
- Número de sinais associados aos RSU.
- Número de sinais por tipo de codificação geográfica, absoluta ou delta.

Na Figura 20 é possível analisar os primeiros 4 gráficos mencionados anteriormente.

Figura 20: *Dashboard*

Estes 4 gráficos servem de exemplo para os restantes da "Dashboard". Os dados presentes nos gráficos são dados de teste. Por exemplo, destes gráficos podemos obter informações de quantos sinais existem, quais as categorias dos sinais existentes e até número de zonas nos sinais.

4.1.8 *IVI Map*

O grupo de páginas "IVI Map"engloba as funcionalidades mais importantes da plataforma. É neste grupo que podemos aceder à página "Editor"onde é possível visualizar o mapa que permite fazer as seguintes funcionalidades no caso do utilizador ter permissões para tal:

- CRUD de sinais.
- CRUD de RSU.

- Atribuir sinais a RSU.
- CRUD de zonas de sinais.
- CRUD de atributos adicionais de sinais.
- Esconder/mostrar áreas de atuação de RSU.
- Esconder/mostrar as zonas dos sinais.
- Esconder/mostrar ligações entre sinais e RSU.
- Bloquear um sinal para que este não possa ser editado acidentalmente.

A Figura 21 representa o mapa com um sinal rodoviário de passagem para peões selecionado. Quando o sinal está selecionado permite ver as zonas que lhe estão atribuídas, neste caso o sinal tem dois trajetos possíveis.

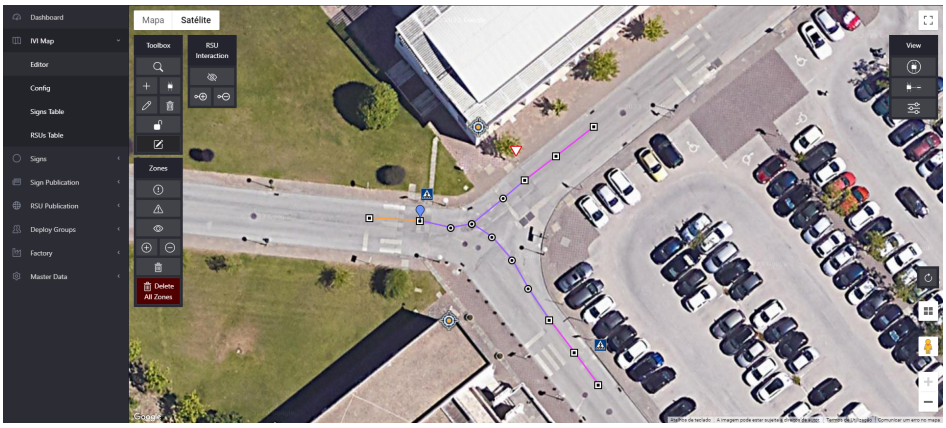


Figura 21: Mapa — Sinal Selecionado

Do lado esquerdo temos o menu lateral, anteriormente mencionado. Do lado esquerdo, mas na janela do mapa, é possível ver as *toolboxs* que permitem realizar as funcionalidades mencionadas anteriormente.

A Figura 22 representa um RSU selecionado. Quando um RSU está selecionado permite conhecer o alcance do transmissor e ver quais são os sinais associados ao RSU.

As cores que podem ser observadas nas zonas dos sinais, linhas de associação e a área de atuação dos RSU são customizáveis. O que permite cada utilizador adaptar-se mais rapidamente à ferramenta, ou para qualquer ajuste que o mesmo necessite. Estas customizações estão disponíveis na página "Config".

A página "Signs Table" permite visualizar todos os sinais inseridos no mapa e serve essencialmente para efetuar pesquisas rápidas de sinais. Por fim, a página "RSUs Table" possibilita a visualização e pesquisa de todos os RSU presentes na plataforma.

Figura 22: Mapa — RSU Selecionado

4.1.9 *Signs*

A opção "Signs" do menu da aplicação, é um grupo com as páginas que permitem efetuar a gestão dos sinais que podem ser adicionados no mapa. Este grupo é dividido pela gestão das categorias e subcategorias dos sinais e pelos *standards* ISO e Vienna.

Dentro deste grupo é possível consultar a página "View" que permite ver todos os sinais do *standard* Vienna e efetuar pesquisas por esta tipologia de sinais. No mesmo grupo de páginas é possível consultar a página "Add", que possibilita a inserção de sinais.

Além da componente de "Vienna", existe a componente "ISO". Esta é parecida com a que já foi mencionada, mas a estrutura para inserção de sinais rodoviários na aplicação é diferente. Também têm uma página para visualizar os sinais e outra para adicionar novos. A Figura 23 representa a adição de um sinal do *standard* ISO.

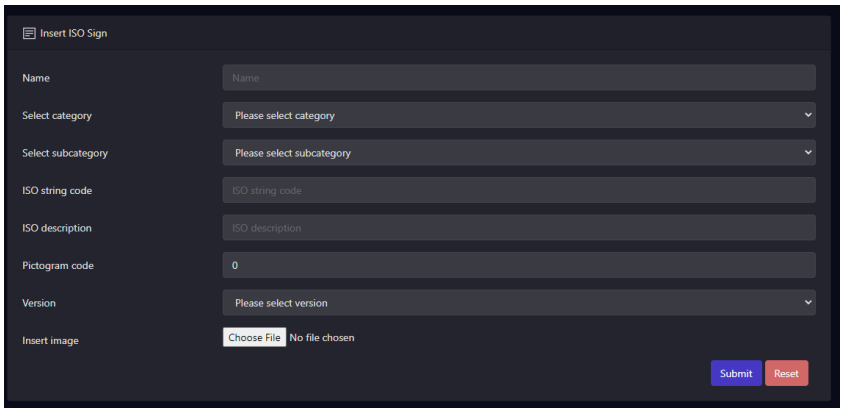


Figura 23: Inserção de um sinal ISO

Como é possível verificar na Figura 23, na criação de um novo sinal do *standard* ISO é necessário inserir os campos *ISO string code* e *ISO description*. Estes campos apenas fazem parte *standard* ISO.

Por fim, o grupo "General Properties" tem dois separadores que permitem aceder às páginas "Categories" e "Subcategories". Estas páginas possibilitam gerir as categorias e as subcategorias dos sinais ISO e Vienna que podemos inserir, e posteriormente implementar no mapa.

4.1.10 *Sign e RSU publication*

As páginas "Sign Publication" e a "RSU Publication" apresentam os sinais e os RSU inseridos no mapa, que estão implementados no terreno.

Estas páginas servem para que os gestores da plataforma consigam entender, verificar e estipular trabalho para outros colaboradores. Na Figura 24 é possível visualizar a página "Sign Publication".

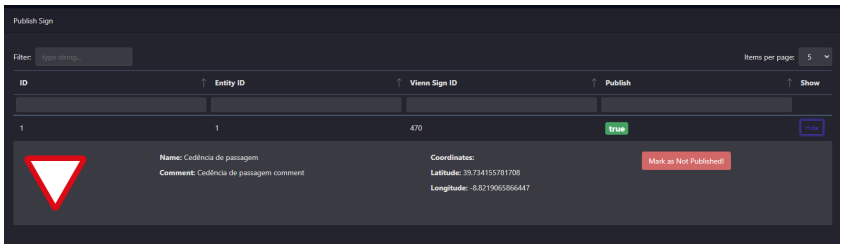


Figura 24: Página "Sign Publication"

Na Figura 24 está representado um sinal que está como "*published*", ou seja, está descrito como implementado em terreno. Caso o gestor necessite pode reverter o estado do sinal para não "*published*".

Os sinais e os RSU apenas ficam disponíveis nestas páginas quando os *workers* concluem o seu trabalho de colocar os sinais e RSU em ambiente real e dão a tarefa como concluída na plataforma.

4.1.11 *Deploy Groups*

A opção do menu "Deploy Groups" é dividida nas seguintes duas páginas. Uma página parar criar os grupos para desenvolvimento e a outra para consulta dos mesmos.

Para criar um grupo é necessário dar um nome e uma descrição ao grupo, para que os *workers* entendam o que é pretendido, quais os sinais do mapa pedidos para

serem colocados no terreno e por fim, quais os *workers* que ficam responsáveis por esta tarefa.

Sempre que um novo grupo é criado pelos gestores, este é inserido numa página a que apenas os *workers* têm acesso. Essa página serve para os *workers* saberem o que têm para fazer. O grupo é inserido com as mesmas informações, lista de sinais e de *workers* atribuídos.

No perfil de gestores existe uma página que permite consultar os grupos criados, o estado da tarefa, quais são os sinais e quem está responsável pela atividade. Não é possível dar as tarefas do grupo como terminadas, dado que esta funcionalidade pertence apenas aos *workers*.

4.1.12 *Factory*

Quando um sinal é criado no mapa, é também criado numa tabela da página "Factory". Esta tabela é responsável por acompanhar o desenvolvimento do sinal.

Um sinal para ser implementado no terreno tem de estar como *Made by Factory*, sendo controlado nesta tabela. Os sinais desta tabela apenas podem ter dois estados, construídos ou não construídos.

Apesar de ser possível criar um grupo para implementar o sinal em terreno sem o sinal estar construído, os *workers* não conseguem colocar o grupo como finalizado se os sinais ainda não estiverem registados como construídos.

Os sinais só saem desta tabela de construídos ou não construídos quando os *workers* colocarem o sinal como implementado no terreno. Desta forma, é necessário existir aprovação do gestor, colocando o sinal como construído e da parte do *worker* que confirma que o sinal está implementado em terreno.

4.1.13 *Master Data*

A opção "Master Data"do menu está dividida em 3 componentes: utilizadores, *roles* e entidades. Essas funcionalidades têm um caráter mais abrangente na administração da plataforma.

Sobre os utilizadores existem duas páginas distintas, uma para criar os utilizadores da plataforma e a outra para consultar os utilizadores criados.

Na inserção de um utilizador é necessário: *username*, endereço de correio eletrónico, número de telemóvel, entidade a que o utilizador estará associado e qual a *role* do utilizador.

Assim que o utilizador é inserido na plataforma, recebe um *email* de boas-vindas e com algumas instruções para a primeira utilização na plataforma. Este *email* permite verificar o endereço de correio eletrónico e contém a palavra-passe inicial para a autenticação na plataforma. O utilizador deverá alterar a sua palavra-passe ou gerar uma nova, através da funcionalidade de recuperar palavra-passe.

A plataforma tem uma página para a consulta de dados dos utilizadores. Esta página oferece vários filtros para consultas personalizadas e rápidas.

Relativamente à componente das *roles*, apenas é permitido consultar as *roles* disponíveis para atribuir aos utilizadores.

Por fim, a componente entidades permite a inserção de novas entidades, inserindo o nome, rua, telefone ou telemóvel e permite opcionalmente a inserção de uma imagem. A inserção de entidades permite que ao criar um utilizador possa ser atribuída a nova entidade. Para além da possibilidade de inserir entidades, também é possível consultar as entidades existentes na plataforma.

4.1.14 *Profile*

O perfil do utilizador permite consultar dados pessoais e dispõem de outras funcionalidades. É nesta página que o utilizador pode escolher e configurar se quer utilizar uma aplicação como o Google Authenticator[82] ou o Microsoft Authenticator[83] como segundo fator de autenticação.

Quando o utilizador quer alterar a sua palavra-passe, sem recorrer à funcionalidade de recuperar palavra-passe, pode fazê-lo nesta página, inserindo a palavra-passe atual e colocar a nova palavra-passe duas vezes.

Na Figura 25 é apresentada a página do perfil de utilizador. Neste exemplo em específico é possível perceber que este utilizador utiliza o endereço de correio eletrónico como segundo fator de autenticação.

Do lado direito da Figura 25 é apresentada a forma de o utilizador configurar o segundo fator de autenticação via aplicação móvel.

4.2 VULNERABILIDADES

Nesta secção, são abordadas as práticas adotadas para desenvolver a aplicação de forma segura, alinhando-se com as diretrizes do OWASP *top* 10 de 2021. O foco é garantir não apenas a funcionalidade da aplicação, mas também a segurança em todos os aspetos.

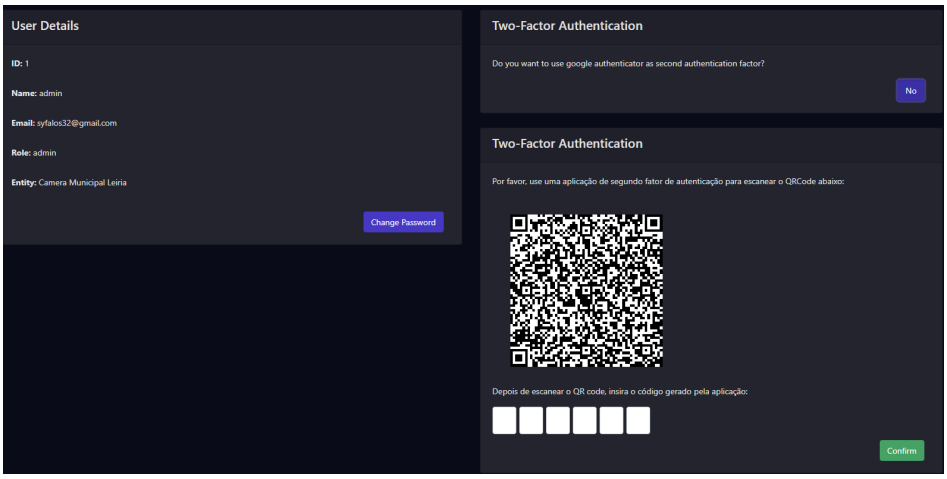


Figura 25: Perfil do utilizador, com opções ativas

É explorado como integrar requisitos de segurança desde a conceção, considerando as boas práticas de desenvolvimento seguro e as peculiaridades do ambiente ITS. É realizado uma análise das vulnerabilidades comuns utilizando o OWASP *top* 10 como guia abrangente.

4.2.1 *Broken Access Control*

Para proteger a aplicação contra ataques provenientes da vulnerabilidade de *broken access control* as restrições de acesso devem ser implementadas corretamente, não permitindo que os utilizadores não autenticados acedam a recursos restritos ou executem ações além das suas permissões.

Em exceção à página de autenticação e ao mecanismo de recuperação da palavra passe, qualquer interação na aplicação *web* necessita de autenticação.

Para proteger a aplicação contra ataques de *brute force*, é recomendado a implementação de uma solução baseada em bloqueio de IP ou em *captchas*. Apesar do uso de *captchas* ser bastante eficiente contra ataques de *brute force* torna o processo de acesso à aplicação inconveniente para o utilizador. Portanto, foi escolhido desenvolver um mecanismo de bloqueio de IP.

O Laravel Sanctum não possui uma funcionalidade integrada para o controlo de tentativas de autenticação com *timeout*. Consequentemente, para a aplicação possuir controlo de tentativas de autenticação, esta funcionalidade teve de ser desenvolvida manualmente.

Quando um utilizador falha a autenticação por 5 vezes tem de esperar 5 minutos para tentar autenticar-se novamente. Se o utilizador falhar a autenticação por 20

vezes no mesmo dia, o seu endereço IP é bloqueado por 24 horas, e a autenticação fica indisponível durante esse período.

Existem aplicações que bloqueiam o endereço de correio eletrónico que tenta autenticar-se sem sucesso. Isto é uma má solução porque, desta forma, qualquer utilizador mal-intencionado pode bloquear as contas de outros utilizadores, assim, pode causar interrupção no funcionamento da organização, bloqueando inúmeras contas e, por sua vez, sobrecarregar os serviços de apoio aos clientes com pedidos de desbloqueio.

No caso do utilizador ter sucesso na autenticação é enviado um código para o seu endereço de correio eletrónico e é redirecionado para uma página de verificação, na qual o utilizador tem de inserir o código para ter acesso à aplicação *web*. O código é gerado de forma aleatória com 10 minutos de validade, reduzindo a possibilidade de ataques de *brute force*.

O processo descrito serve como segundo fator de autenticação, garante que apenas os utilizadores que possuem acesso ao endereço de correio eletrónico registado no sistema completem o processo de autenticação. A página de verificação é mostrada na Figura 26.

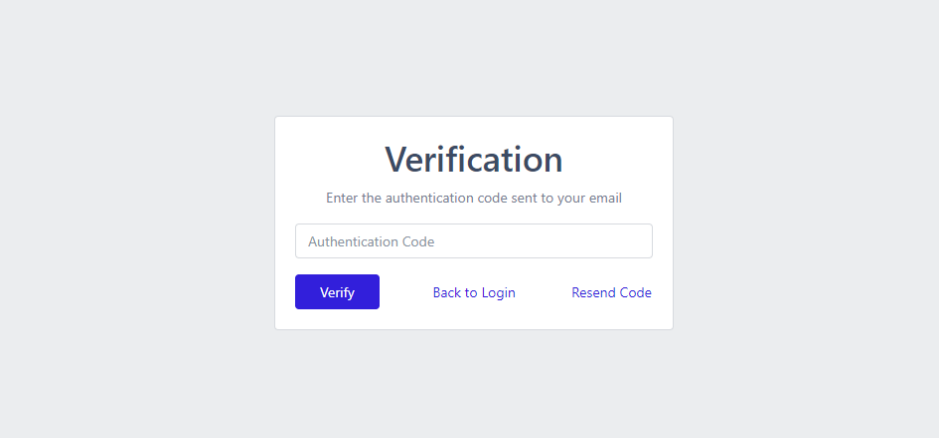


Figura 26: Página de verificação

A aplicação cria um *token* de autenticação quando o utilizador se autêntica na aplicação. O *token* garante que apenas os utilizadores autenticados possam realizar pedidos à API. O *token* é enviado no cabeçalho em *Authorization* como *Bearer token*.

A aplicação desenvolvida permite restringir o acesso aos recursos da aplicação conforme as funções e responsabilidades de cada utilizador. Ao implementar diferentes tipos de utilizadores com permissões e acessos específicos, é possível limitar o risco de um ataque de *broken access control*.

A criação deste *token* de autenticação é criado através do pacote Laravel Sanctum. Este pacote fornece autenticação baseada em *tokens* para aplicações.

Inicialmente o *token* foi criado com o JWT, mas para esta aplicação faz mais sentido utilizar apenas a criação de *tokens* através do Laravel Sanctum. O JWT é mais utilizado para casos de troca de informações entre serviços diferentes. O Laravel Sanctum garante os recursos de validação e revogação de *tokens*.

A aplicação é composta por 6 perfis de utilizador:

- Administrador: Utilizador com acesso a todos os pedidos da aplicação, esta *role* é demasiado poderosa e requer cautela ao atribuí-la a um utilizador.
- Administrador de entidade: Apesar de ser um administrador, o seu acesso aos recursos do sistema são limitados. Tem acesso de CRUD aos grupos que associam os sinais e pode consultar os sinais e RSU criados no sistema.
- *Planner*: Utilizador responsável pelo CRUD relacionado com os objetos do mapa, apenas tem acesso ao mapa e a páginas com as vistas das tabelas de sinais e RSU.
- Fábrica: Utilizador que apenas consegue consultar os sinais marcados como "*Signs to Make*".
- Gestor de implementação: Permite realizar os CRUD relacionados com o mapa, página de sinais, RSU e grupos de implementação.
- Trabalhador: Funcionário que anda no terreno a implementar os sinais e RSU. Apenas têm acesso ao mapa para visualizar os sinais e RSU. Também têm acesso a ver os seus grupos de implementação.

Quando um utilizador não tem acesso a uma determinada página recebe a página de erro com o código 403. A Figura 27 mostra a página de erro. A receção desta página deve ser interpretada como que o servidor entendeu a solicitação, mas recusa a autorizá-la. Este erro é conhecido por *Forbidden*.

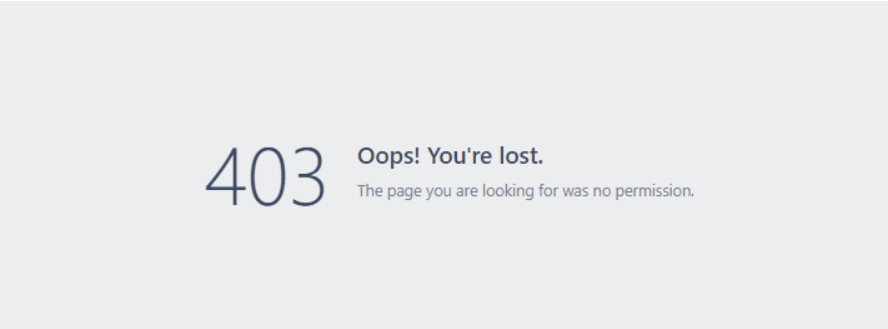


Figura 27: Sem permissão para aceder a uma determinada página

Os pedidos que o utilizador faz sem ter a respetiva autorização, o servidor responde com a mensagem de erro apresentada na Figura 28.

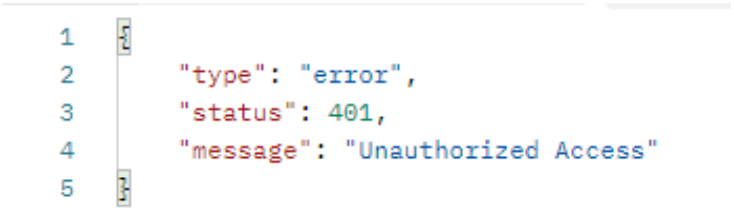


Figura 28: Resposta do servidor para um pedido em autorização

4.2.2 *Cryptographic Failures*

Os dados transmitidos entre o servidor e o cliente devem ser criptografados utilizando *Transport Layer Security* (TLS)/*Secure Sockets Layer* (SSL). O Laravel tem suporte para TLS/SSL ativado por padrão, o que significa que o tráfego entre o servidor e o cliente é criptografado por padrão. Apesar de ainda não ser relevante visto que aplicação está alojada como HTTP.

Utilizar criptografia forte para proteger dados confidenciais é outra medida importante para evitar ataques prevenientes de vulnerabilidades relacionados com *Cryptographic Failures*. O Laravel oferece suporte para vários algoritmos de criptografia como *Advanced Encryption Standard* (AES), *Blowfish* e outros. Neste projeto para proteger dados confidenciais foi utilizado o método `Hash::make()`. É uma função *built-in* do Laravel que permite criar *hashes* de senhas utilizando um algoritmo de *hash* forte e seguro. Este método é uma forma segura de proteger as senhas dos utilizadores na aplicação.

O Laravel utiliza o algoritmo de *hash Bcrypt* por padrão. O *Bcrypt* é um algoritmo de *hash* seguro e robusto, desenvolvido para resistir a ataques de força bruta e criptográficos. Além disso, o Laravel utiliza um valor aleatório e único para cada *hash* de senha, tornando as *hashes* mais seguras.

Utilizar um *token* de autenticação, como o *Bearer Token*, pode ajudar a prevenir ataques de criptografia, como a intercetação de dados em trânsito ou a injeção de código malicioso em solicitações.

4.2.3 *Injection*

A aplicação contém muitas inserções em formulários ou outros campos que interagem com a base de dados, portanto é importante validar adequadamente a entrada de dados e restringir o acesso dos utilizadores aos recursos do sistema.

Todos os pedidos realizados pelos clientes da aplicação são enviados pela API para o *backend*. Como referido anteriormente, o servidor não comunica diretamente com a base de dados. Como é utilizada a *framework* Laravel, também é utilizado o Eloquent para todas as comunicações com a base de dados através dos modelos da aplicação.

Apesar do Eloquent ser suficiente para a grande maioria dos ataques de SQL *Injection* foram tomadas outras medidas contra este tipo de ataques. Por exemplo, nos formulários houve atenção nas seguintes medidas:

1. Utilizar os atributos dos valores esperados, como *'type="email"'* no formulário HTML ajuda a garantir que apenas endereços de correio eletrónicos válidos sejam inseridos, impedindo a inserção de código malicioso naquele campo.
2. Utilização de mensagens de erro genéricas para não expor informações sensíveis que poderiam, por exemplo, causar ataques de *brute force*, apresentado na Figura 29. Existem aplicações que dão a entender que o endereço elétrico já existe no sistema, isso é considerado uma falha de segurança.

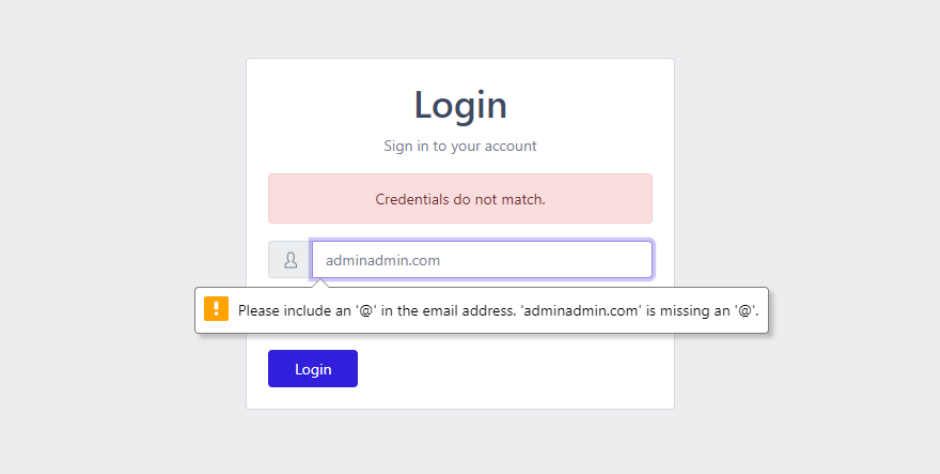


Figura 29: Mensagens de erro

Estas foram as medidas utilizadas em todos os *inputs* dos utilizadores na aplicação. A nível do cliente foi restringido a inserção de determinados dados e a nível de servidor foi utilizado o Eloquent para todos os pedidos da aplicação de forma correta e segura.

No OWASP 2021 a vulnerabilidade XSS passou a ser englobada pela vulnerabilidade de *injection*. Esta vulnerabilidade é ilegível nesta aplicação, os parâmetros são cuidadosamente controlados, e todas as entradas de utilizador são devidamente validadas para prevenir a injeção de *scripts* maliciosos. A aplicação também emprega *Content Security Policy* (CSP) para restringir as fontes de recursos permitidas, e os cabeçalhos de segurança apropriados são configurados no servidor para fornecer camadas adicionais de proteção.

4.2.4 *Insecure Design*

Uma das medidas tomada para mitigar vulnerabilidades provenientes de *insecure design* é a utilização de CORS. CORS é uma política de segurança que impede que solicitações sejam feitas a partir de um *website* não autorizado. Desta forma a aplicação é protegida contra solicitações não autorizadas.

Além da utilização do CORS existiu, um cuidado em evitar a exposição excessiva de informações sobre os detalhes de erros, a qual poderia ser utilizada por um atacante de forma maliciosa.

4.2.5 *Security Misconfiguration*

A falta de configuração segura ocorre quando os sistemas são configurados de maneira inadequada, tais como configurações padrão inseguras, informações sensíveis expostas ou serviços desnecessários habilitados, facilitando ataques e explorações.

A vulnerabilidade de *Security Misconfiguration* é muito recorrente em aplicações que recorrem a ficheiros xml. A aplicação *web* descrita neste documento não interage com ficheiros xml. Todos os *inputs* que permitem inserir ficheiros têm validações para restringir aos formatos de ficheiros permitidos.

Foi importante verificar os pacotes que estão em uso na aplicação e utilizar apenas os necessários. Assim como verificar os erros que a aplicação envia para o utilizador, para não mostrar informações sensíveis. Algumas medidas tomadas:

- Eliminar todos os *console.log()* que serviram de apoio ao desenvolvimento da aplicação.
- Verificar se existem erros a redirecionar para páginas de configuração da aplicação.
- Desativar o modo *debug* da aplicação.

4.2.6 *Vulnerable and Outdated Components*

Uma forma de mitigar vulnerabilidades provenientes de *vulnerable and outdated components* é atualizar devidamente as bibliotecas, *frameworks* ou *plugins* utilizados pela aplicação isto porque as versões desatualizadas podem conter falhas conhecidas e serem alvos fáceis para os invasores.

Não basta atualizar os diferentes componentes utilizados no projeto, é necessário analisar as diferenças entre versões e verificar as compatibilidades com os restantes componentes. Em alguns casos existem funções classificadas como *deprecated*, as quais exigem alterações no código.

A Listagem 1 mostra as versões da aplicação antes de serem atualizadas e a Listagem 2 mostra as versões atuais da aplicação.

Listagem 1: Versões antigas

```
"require": {
    "php": "^7.3",
    "fideloper/proxy": "^4.4",
    "fruitcake/laravel-cors": "^2.0",
    "guzzlehttp/guzzle": "^7.0.1",
    "laravel/framework": "^8.40",
    "laravel/tinker": "^2.5",
    "laravel/ui": "^3.3",
    "spatie/laravel-medialibrary": "^9.6",
    "spatie/laravel-permission": "^4.0",
    "tymon/jwt-auth": "^1.0"
},
"require-dev": {
    "facade/ignition": "^2.5",
    "fakerphp/faker": "^1.9.1",
    "laravel/sail": "^1.0.1",
    "mockery/mockery": "^1.4.2",
    "nunomaduro/collision": "^5.0",
    "phpunit/phpunit": "^9.3.3"
},
```

Listagem 2: Versões atuais

```
"require": {
    "php": "^8.1",
    "doctrine/dbal": "^3.0",
    "guzzlehttp/guzzle": "^7.0.1",
    "laravel/framework": "^10.0.0",
    "laravel/sanctum": "^3.2",
    "laravel/tinker": "^2.7",
    "spatie/laravel-medialibrary": "^10.0.0",
    "spatie/laravel-permission": "^6.0"
},
"require-dev": {
    "spatie/laravel-ignition": "^2.0",
    "fakerphp/faker": "^1.9.1",
    "phpunit/phpunit": "^10.0",
    "kitloong/laravel-migrations-generator": "^6.0",
    "laracademy/generators": "^3.6",
    "laravel/sail": "^1.0.1",
    "nunomaduro/collision": "^7.0"
},
```

Como é possível confirmar existem pacotes atualizados, removidos e novos no projeto. Inicialmente foi necessário perceber o que mudou da versão 8.40 da *framework* Laravel para a versão 10.

Uma das alterações verificadas foi o Laravel 10 tem um *Middleware* CORS. Sendo assim, o pacote "fruitcake/laravel-cors" deixa de ser necessário no projeto. Por esta razão, o pacote foi removido do ficheiro "composer.json" sendo colocado o *Middleware HandleCors* através do ficheiro "Kernel.php". Estas alterações são provenientes da documentação oficial do Laravel, do próprio pacote e do *website* Packagist[84].

Outro pacote removido foi o "fideloper/proxy", este pacote foi incorporado ao núcleo do Laravel.

O pacote "laravel/ui", apesar de fazer sentido neste projeto, nunca era utilizado, por isso o pacote foi removido do projeto. Outro pacote que nunca foi utilizado é o "mockery/mockery", por isso foi removido.

Como referido anteriormente, foi analisada a forma mais correta de criar os *tokens* de autenticação para este projeto. Como se trata de uma *Single-Page Application* (SPA) não faz sentido estar a utilizar JWT *tokens*. Portanto, foi removido o pacote "tymon/jwt-auth" e instalado o "laravel/sanctum", o qual cria os *tokens* de forma mais automatizada.

O pacote "facade/ignition" era utilizado nas versões anteriores ao Laravel 8.0. Nas versões mais atuais é utilizado o pacote "spatie/laravel-ignition". Estes pacotes "Ignition" são uma página de erro personalizável para aplicações desenvolvidas com Laravel.

O “kitloong/laravel-migrations-generator” é um pacote adicionado ao projeto para gerar migrações do Laravel a partir de uma base de dados existente, incluindo índices e chaves estrangeiras. Outro pacote adicionado foi o “laracademy/generators” que oferece um conjunto de ferramentas, as quais ajudam a acelerar o processo de desenvolvimento de uma aplicação Laravel.

O NPM identifica as vulnerabilidades nos pacotes utilizados pela aplicação. A Figura 30 mostra a quantidade de vulnerabilidades a aplicação tinha.

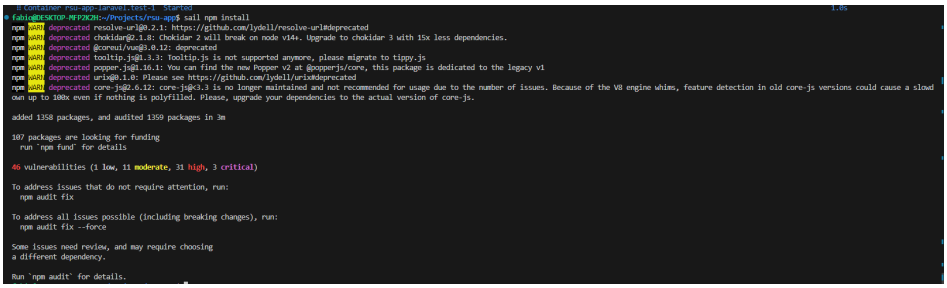


Figura 30: Vulnerabilidades na aplicação

Para resolver as vulnerabilidades sem impactar o projeto foi utilizado o comando ’*sail npm audit fix*’. Este comando procura por vulnerabilidades conhecidas nos pacotes do projeto NPM e tenta corrigi-las automaticamente, quando possível, atualizando as dependências para versões mais recentes e seguras. A Figura 31 mostra a execução do comando ’*sail npm audit fix*’.

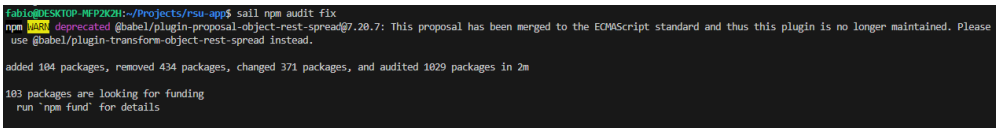


Figura 31: Resolução das vulnerabilidades

Após a execução do comando, foi necessário atualizar manualmente o *vue-loader*. Esta atualização permitiu correr novamente o comando ’*sail npm audit fix*’ e assim atualizar mais pacotes. Com esta ação, a aplicação ficou com 12 vulnerabilidades, tendo uma melhoria de 34 vulnerabilidades resolvidas e não foram encontradas vulnerabilidades críticas.

4.2.7 *Identification and Authentication Failures*

Esta categoria aborda os problemas relacionados à autenticação, como *passwords* fracas, ausência de mecanismos de autenticação adequados, que podem permitir o acesso não autorizado ao sistema e a dados sensíveis.

Uma das primeiras preocupações é não permitir a criação de contas de utilizador com *passwords* fracas. Desta forma, é mitigada a utilização de senhas fracas por parte dos utilizadores e, assim, são minimizados os problemas com a autenticação.

Segundo o OWASP, uma *password* robusta deve ter no mínimo 8 caracteres, mas recomenda que o comprimento mínimo seja de pelo menos 12 caracteres. O OWASP também incentiva a utilização pelo menos letras maiúscula, minúscula, um número e um caractere especial.

Para garantir que o utilizador cria uma *password* com estes requisitos é necessário efetuar a respetiva validação. Para fazer esta validação foi utilizado o *regex*, mais especificamente o *website regular expressions 101* [85], o qual auxilia neste processo.

A expressão utilizada foi a seguinte:

```
/^(?=.*?\d)(?=.*?\p{Ll})(?=.*?\p{Lu})(?!.*?\s)(?=.*?[^\p{L}\d])/
```

A expressão regular possui os seguintes elementos:

- Os pontos `('.')` corresponde a qualquer caractere, exceto para terminadores de linha `('\n')`.
- O `('*?')` corresponde ao *token* anterior entre zero e um número ilimitado de vezes, o menor número de vezes possível, expandindo conforme necessário.
- O `('\d')` corresponde a um dígito (equivalente a [0-9]).
- O `('\p{Ll}')` corresponde a uma letra minúscula com uma variante maiúscula.
- O `('\p{Lu}')` corresponde a uma letra maiúscula com uma variante minúscula.
- O `('\s')` corresponde a qualquer caractere de espaço em branco (equivalente a `('\r\n\t\f\v')`.
- O `('\p{L}\d')` obriga que existe pelo menos um tipo de letra de qualquer idioma e um dígito (equivalente a [0-9]).
- O `('?=')` é utilizado para indicar que a expressão deve conter essa regra.
- O `('?!')` é utilizado para indicar que a expressão não pode conter essa regra.

Desta forma é garantido que todos os utilizadores da aplicação utilizam *passwords* robustas. Para além desta validação, ainda é obrigatório que a *password* seja composta por 12 caracteres. Na Figura 32 podemos verificar o formulário com dados preenchidos, mas não correspondentes às validações.

Além de requer todos os utilizadores tenham *passwords* robustas, a implementação da segurança estende-se à utilização de *Second Factor Authentication* (2FA). Os utilizadores podem escolher entre dois métodos de 2FA, através do endereço de

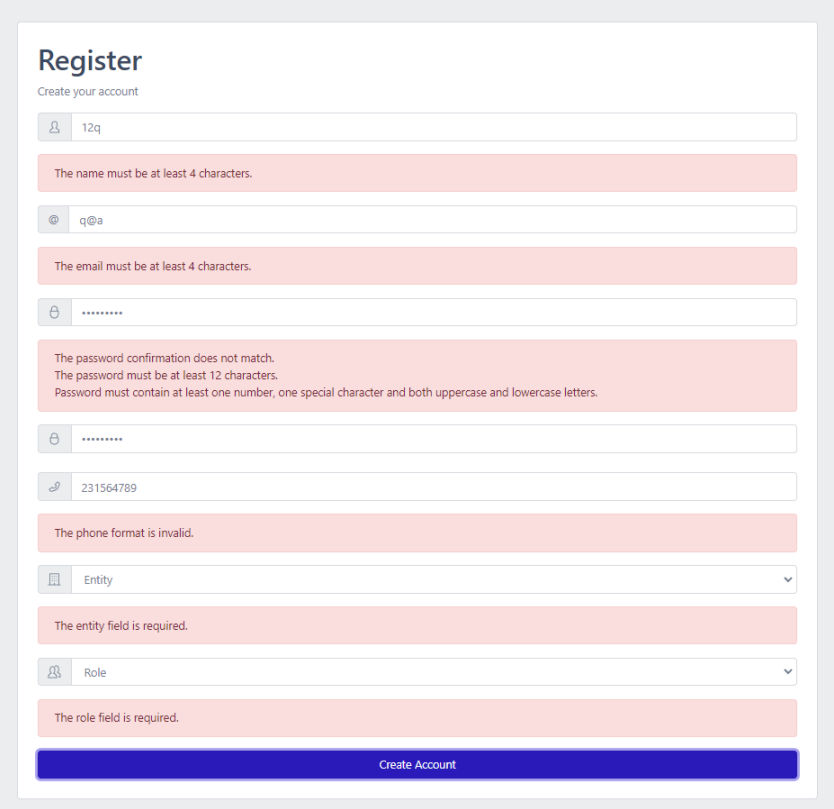


Figura 32: Formulário de registo

correio eletrónico ou por uma aplicação móvel, como o Google Authenticator [82] ou o Microsoft Authenticator [83].

4.2.8 *Software and Data Integrity Failures*

Esta vulnerabilidade está relacionada a problemas que podem resultar na modificação ou corrupção não autorizada de *software* ou dados da aplicação. Uma forma de proteger contra esta vulnerabilidade é efetuar a validação das *cookies* utilizadas na aplicação.

Por norma, esta vulnerabilidade está disponível quando o cliente consegue comunicar diretamente com a base de dados, o que permite ao utilizador criar uma *payload* que permita a alteração de campos da base de dados. Neste caso os clientes da aplicação não têm acesso direto à base de dados, portanto a aplicação está protegida neste contexto.

O Laravel utiliza criptografia para *deserialization* da informação com uma chave. Em princípio, isto é suficientemente seguro para não permitir ataques de *insecure deserialization*. Uma das vulnerabilidades do *Hypertext Preprocessor* (PHP) é a incorreta utilização das chamadas *magic functions*, como o *__construct* e *__destruct*.

Este tipo de funções possibilita a utilização da função *inserialize*, a qual permite ao invasor converter dados serializados, como *strings*, de volta para seu formato original de objeto ou estrutura de dados.

4.2.9 *Security Logging and Monitoring Failures*

O Laravel oferece um sistema de *logs* integrado que permite registar eventos, erros e informações importantes durante a execução da aplicação. O sistema de *logs* permite monitorizar o comportamento da aplicação, diagnosticar problemas e investigar atividades específicas. Os *logs* facilitam a identificação e resolução de problemas em ambientes de produção, ajudando a entender o fluxo de execução, identificar erros e depurar problemas de desempenho. Além disso, os *logs* são essenciais para investigar e analisar eventos importantes, como atividades de utilizadores, erros críticos e transações do sistema, o que pode ser crucial para garantir a segurança e a estabilidade da aplicação.

Foram criados canais de *logs* personalizados para assegurar a distinção entre os diferentes tipos de *logs*. Ter *logs* sobre a autenticação foi uma preocupação, para controlar diversos tipos de ataques. Na Figura 33 está apresentado os *logs* sobre tentativas falhadas de autenticação.

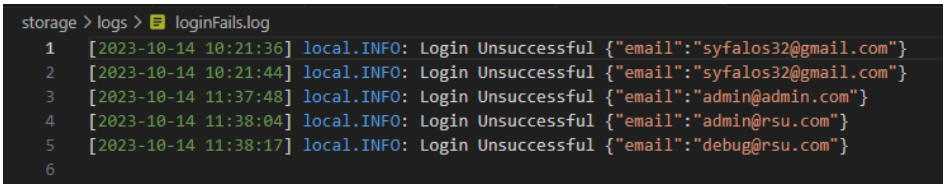


Figura 33: *Logs* de falhas de autenticação

Adicionalmente aos *logs* de falhas de autenticação, também foram criados *logs* para gestão de utilizadores, processos de administradores e autenticações com sucesso. Estes ficheiros ficam armazenados apenas no servidor, mas o ideal seria estes serem enviados para um SIEM.

4.2.10 *SSRF*

O desenvolvimento da aplicação com a *framework* Laravel, permitiu implementar diversas medidas para mitigar potenciais vulnerabilidades, incluindo SSRF. A aplicação incorpora validações rigorosas de entrada para prevenir manipulações maliciosas de URL. Além disso, foram adotadas práticas de codificação segura para URL, evitando assim possíveis ataques.

A inclusão de padrão de proteções CSRF no Laravel oferece uma camada adicional de segurança contra ameaças relacionadas a solicitações não autorizadas. Ressalto que a segurança efetiva da aplicação resulta de uma abordagem abrangente, incluindo configurações de servidor seguras, boas práticas de desenvolvimento e atualizações regulares do Laravel e das suas dependências.

4.3 SÍNTESE

Neste capitulo foram detalhadas as funcionalidades da aplicação, alinhadas aos princípios do ITS. Foram abordados os mecanismos e funcionalidades incorporadas para proteger a aplicação seguindo as melhores práticas do OWASP *top* 10.

No próximo capítulo (5), é descrito os testes executados do PenTest neste projeto, seguindo a metodologia PTES e a *checklist* do OWASP WSTG. É mencionado o processo para procurar as vulnerabilidades, assim como para mitigar as mesmas.

5

PENTEST

No capítulo de PenTest é detalhado o teste de penetração realizado, adotando a metodologia PTES e guiados pela *checklist* abrangente do OWASP WSTG. Conduzindo a uma análise meticulosa da resistência da aplicação a possíveis ameaças, este processo minucioso não apenas revelou potenciais vulnerabilidades, mas também permitiu a implementação de correções.

Ao explorar as complexidades da segurança, enfrentamos diversas ameaças em potencial, garantindo não apenas a funcionalidade robusta da aplicação, mas também a sua resiliência diante de possíveis ataques. Este enfoque proativo reforça não apenas os resultados obtidos, mas também a postura contínua de segurança incorporada ao desenvolvimento da aplicação.

5.1 *PRE-ENGAGEMENT*

Na fase inicial de *Pre-Engagement*, é essencial delinear claramente o âmbito do teste de penetração (PenTest). Em certos cenários, como no projeto em questão, sem uma distinção clara entre o papel do proprietário da aplicação (cliente) e o *tester* do PenTest, é crucial estabelecer protocolos para garantir a imparcialidade e a objetividade do PenTest. Situações semelhantes podem ocorrer em organizações que possuem equipas de desenvolvimento e de PenTest internas.

Para este projeto específico, foi escolhida uma abordagem de *grey-box* para simular um ataque de um invasor externo. No entanto, em cenários no qual o *pentester* encontra obstáculos e não consegue progredir de forma independente, serão fornecidas informações relevantes para facilitar o avanço do PenTest, garantindo assim a eficácia do processo.

O âmbito do PenTest é restrito à avaliação da aplicação *web*, sem incluir a análise do servidor onde a aplicação *web* está hospedado ou o acesso ao código-fonte da aplicação. No entanto, se o código-fonte estiver disponível, o *pentester* poderá utilizá-lo para identificar as respetivas vulnerabilidades, replicando assim as táticas de um invasor. É importante observar que o servidor não será testado para evitar interrupções e limitações de acesso.

Ao delinear claramente as restrições e limitações do PenTest, é possível garantir que o processo seja conduzido de maneira eficaz e ética, sem comprometer a estabilidade ou a segurança da infraestrutura existente. A definição cuidadosa do âmbito do PenTest é fundamental para garantir a obtenção de resultados significativos e relevantes para a avaliação da postura de segurança da aplicação *web*. Para efeitos de testes foi fornecida uma conta com a *role* de administrador para efetuar o PenTest.

5.2 *INTELLIGENCE GATHERING*

Esta secção é baseada nas tarefas de *Information Gathering* da *checklist* do OWASP WSTG. Este procedimento também é conhecido como enumeração e está dividido em 10 tarefas. De seguida é apresentado o desenvolvimento da enumeração com base nas 10 tarefas da *checklist*.

Conforme a primeira tarefa da *checklist* é necessário obter o máximo de informação sobre o domínio externamente, via navegadores *web* e com operadores de pesquisa. No entanto, a aplicação *web* está hospedado apenas no servidor e apenas é possível aceder através da rede interna ou com uma VPN. Devido a esta condição não foi efetuada nenhuma análise através dos motores de busca, nem foi analisado o histórico da aplicação.

Continuado a *checklist* é essencial determinar a versão e o tipo de servidor *web* em execução para permitir a descoberta adicional de quaisquer vulnerabilidades conhecidas. Para isso foi utilizado o programa telnet[86] que obteve uma resposta HTTP para a solicitação *GET* enviada ao servidor com um código de status de 200 OK, o que significa que a solicitação foi bem-sucedida. A resposta contém vários cabeçalhos indicando detalhes sobre o conteúdo e o servidor.

- *Cache-Control*: Indica a diretiva de controlo de *cache*, que neste caso é definida como "*no-cache, private*".
- *Content-Type*: Indica o tipo de conteúdo que está a ser retornado, neste caso, "*text/html; charset=UTF-8*".
- *Date*: Indica a data e hora em que a resposta foi gerada.
- *Server*: Indica o *software* do servidor que responde à solicitação, que neste caso é "*Caddy*".
- *Set-Cookie*: Define os *cookies* que serão armazenados no cliente.
- *X-Powered-By*: Indica a tecnologia ou linguagem de programação utilizada para criar o conteúdo, que neste caso é "*PHP/8.1.13*".

- *Transfer-Encoding*: Indica a codificação de transferência do corpo da mensagem, neste caso "*chunked*".

Além dos cabeçalhos referidos, a resposta do comando telnet[86] ainda contém o corpo da resposta, a qual é uma página HTML contendo metadados e *scripts*, bem como referências a recursos externos.

O próximo passo da *checklist* corresponde à identificação de caminhos e funcionalidades por meio de ficheiros de metadados, assim como extrair e mapear outras informações que levem a uma melhor compreensão do sistema em questão.

Um ficheiro importante de analisar é o "robots.txt" porque fornece informações sobre quais áreas de uma aplicação *web* podem ser exploradas e quais devem ser restritas. Porém, a análise do "robôs.txt" não é relevante para nós, pois a aplicação *web* não está hospedada publicamente e por isso este ficheiro não foi configurado.

Dentro deste passo é referido a análise das *META Tags*, estas informações podem ter valor na identificação de tecnologias utilizada e caminhos ou funcionalidade adicionais para explorar e testar. Para as encontrar foi realizado um *wget*[87] ao servidor, o que retornou com a página *index* da aplicação.

Apenas foram identificadas 3 *tags* neste ficheiro, todas pouco relevantes:

- charset=”utf-8": esta *tag* especifica o conjunto de caracteres utilizados.
- http-equiv=”X-UA-Compatible” content="IE=edge”: esta *tag* é utilizada para fornecer instruções especificas para o *browser* sobre a compatibilidade com o Internet Explorer.
- name=”viewport” content=”width=device-width,initial-scale=1.0": esta *tag* permite controlar o comportamento da *viewport* em dispositivos móveis.

Outros ficheiros reconhecidos nos PenTest, tais como o sitesmap, security.txt e humans.txt não foram encontrados nesta aplicação *web*.

Para descobrir possíveis subdomínios da aplicação foram utilizadas 3 ferramentas. Inicialmente foi utilizado o ffuf[88] com uma lista personalizada para procurar subdomínios comuns, retirada do *github* do danielmiessler/SecLists[89]. A segunda ferramenta utilizada foi o Sublist3r[90] que é uma ferramenta de procura de subdomínios que utiliza motores de busca para identificar subdomínios associados a um domínio específico. Por fim, foi utilizado o Subfinder que é outra ferramenta popular utilizada para descobrir subdomínios de um determinado domínio. Ela utiliza uma abordagem abrangente, que inclui verificação de subdomínios em diversos recursos, como bases de dados públicos, registos de DNS e outros repositórios *online*.

Nenhuma destas ferramentas conseguiu encontrar um subdomínio válido, no entanto, o Sunfinder encontrou um suposto subdomínio que é o *www*. Pelas pesquisas realizadas apenas é um indicador de que o servidor está ativo e a responder a pedidos

HTTP ou uma falha de configuração no servidor que poderá ser explorada. Devido a estes resultados foi percebido que os seguintes testes serão todos direcionados apenas ao domínio da aplicação *web*.

Para descobrir rotas, parâmetros, pastas e ficheiros do lado do servidor *web* foi utilizada a ferramenta wfuzz[91] com auxílio de outra lista do *github* do danielmiessler/SecLists[89] direcionada para descoberta de conteúdo na *web*. A Figura 34 mostra os resultados desta ferramenta.

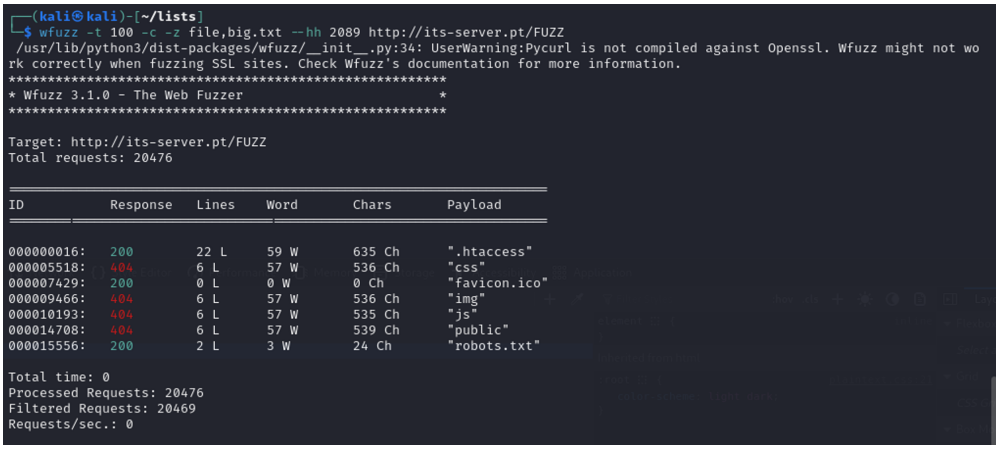


Figura 34: Resultados da ferramenta wfuzz

O que o wfuzz[91] encontrou com o *status code* 404 são pastas existentes na pasta *public*, onde é possível realizar novas pesquisas por ficheiros dentro destas pastas. A pasta *public* do projeto está pública, no entanto, caso um utilizador tente aceder ao seu conteúdo sem ter efetuado o *login* é redirecionado para a página do *log in*. A única forma de aceder aos dados na pasta *public* é através da comunicação do *backend*. Por exemplo, executando o comando "wget http://its-server.pt/img/RSURange.png".

Para confirmação dos resultados obtidos foi utilizado o dirb[92] que conseguiu encontrar 2 novos ficheiros ("index.html"e "web.config") com a lista de palavras comuns para descobrir ficheiros e pastas. Para além destas duas ferramentas foram utilizadas várias listas.

As análises anteriores são consideradas manuais. Para finalizar a recolha de dados ainda foi utilizado o OWASP ZAP[37] para realizar uma análise automática que encontrou um novo ficheiro (sitemap.xml), mas não encontrou as mesmas pastas ("img" e "public").

Um aspeto de segurança importante de identificar no processo de enumeração é as portas abertas no servidor. Para fazer o *scan* das portas foi utilizado o nmap[93], apresentado na Figura 35.

Como é possível confirmar na Figura 35 o *scan* do nmap permitiu identificar as seguintes portas abertas:

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -sV  its-server.pt -v
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-25 12:31 EST
NSE: Loaded 46 scripts for scanning.
Initiating Parallel DNS resolution of 1 host. at 12:31
Completed Parallel DNS resolution of 1 host. at 12:31, 0.00s elapsed
Initiating Connect Scan at 12:31
Scanning its-server.pt (172.22.21.43) [1000 ports]
Discovered open port 3306/tcp on 172.22.21.43
Discovered open port 22/tcp on 172.22.21.43
Discovered open port 80/tcp on 172.22.21.43
Completed Connect Scan at 12:31, 6.38s elapsed (1000 total ports)
Initiating Service scan at 12:31
Scanning 3 services on its-server.pt (172.22.21.43)
Completed Service scan at 12:32, 6.06s elapsed (3 services on 1 host)
NSE: Script scanning 172.22.21.43.
Initiating NSE at 12:32
Completed NSE at 12:32, 0.92s elapsed
Initiating NSE at 12:32
Completed NSE at 12:32, 0.17s elapsed
Nmap scan report for its-server.pt (172.22.21.43)
Host is up (0.031s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    Caddy httpd
3306/tcp open  mysql   MySQL 8.0.31
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.75 seconds
```

Figura 35: Scan nas portas do servidor, utilizando o nmap

- Porta 22: serviço OpenSSH 7.6p1 no sistema Ubuntu Linux.
- Porta 80: serviço de servidor *web* Caddy.
- Porta 3306: serviço MySQL 8.0.31.

Uma forma automatizada para identificar as tecnologias utilizadas passa pelo uso da aplicação Wappalyzer[94]. Esta ferramenta analisa os *scripts* e os metadados presentes na aplicação *web* para identificar as tecnologias da aplicação. A Figura 36 mostra os resultados desta ferramenta na extensão do navegador.

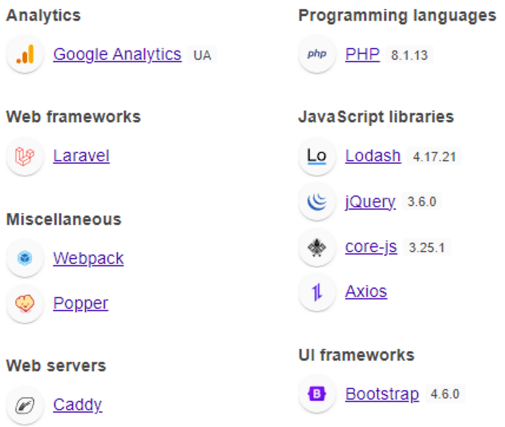


Figura 36: Tecnologias encontradas na extensão do navegador, utilizando Wappalyzer

Pode-se observar que foram encontradas diversas tecnologias mencionadas anteriormente, como a *framework* Laravel, o *web server* Caddy ou até a versão do PHP.

Além do Wappalyzer foi também utilizado o whatweb[95], com o qual se obtiveram resultados semelhantes. Acrescenta-se que esta feramenta detetou também as *cookies* que a aplicação utiliza. A Figura 37 mostra os resultados obtidos.

Figura 37: Tecnologias encontradas na aplicação *web*, utilizando whatweb

As *cookies* identificadas são conhecidas em aplicações *web*. A *cookie XSRF-TOKEN* é utilizada para prevenir ataques de falsificação de solicitações entre *websites* CSRF, fornecendo um mecanismo para validar que as solicitações feitas numa aplicação *web* foram originadas na própria aplicação *web* e não de fontes externas maliciosas.

A *cookie laravel_session* é uma *cookie* utilizada na *framework* Laravel para armazenar a sessão do utilizador. Ela contém informações sobre a sessão específica do utilizador, o que permite que o Laravel identifique e investigue os utilizadores entre diferentes solicitações HTTP, mantendo o estado da sessão e fornecendo uma experiência de utilizador consistente durante a interação com a aplicação Laravel.

5.3 *THREAT MODELLING*

Visando assegurar uma execução precisa de um teste de penetração, esta secção apresenta uma metodologia de análise de ameaças, que inclui a recolha de informações relevantes, a identificação e classificação de ativos primários e secundários.

Inicialmente foi realizado o acesso à aplicação *web* com a conta de administrador fornecida para a realização do PenTest, com o intuito de analisar a informação presente na aplicação *web* e qual o tipo de informação que é possível obter do mesmo. Desta forma, é possível entender o que tem maior risco para o modelo de negócio associado à aplicação.

Com esta conta de administração é possível aceder a dados de colaboradores, tais como, nome, *email*, número telefónico e qual a role desempenhada na aplicação. Este tipo de informação sensível não deve ser acessível por qualquer pessoa, denominada por "*Employee Data*" pela metodologia PTES.

Os colaboradores são um ativo da organização. Dessa forma, esses são os elementos que podem ser utilizados para a disseminação de informações, tais como a

manipulação para a tomada de decisões ou a atuação de forma que cause danos à organização ou a permissão de agressões ainda maiores.

A plataforma não dispõe de dados relacionados com clientes, políticas, procedimentos, *markting*, financeiras, facilitando o processamento de proteção das informações e o cumprimento da legislação.

É crucial que as informações criadas na aplicação sejam fidedignas e precisas, pois estas têm impacto no trabalho futuro de outros colaboradores, como a colocação de sinais rodoviários em terreno. Tais informações não podem ser modificadas ou negligenciadas, garantindo a segurança e a eficácia das operações no ambiente rodoviário. Estas informações são críticas para o modelo de negócio onde se enquadra a aplicação, portanto a análise de vulnerabilidades e a exploração vão ser relacionadas com estas informações.

5.4 VULNERABILITY ANALYSIS

Devido à utilização das *frameworks*, Laravel e Vue.js, existem muitas vulnerabilidades que são mitigadas automaticamente pela estrutura das mesmas. Adicionalmente, estas *frameworks* ainda oferecem diversas soluções para mitigar outras vulnerabilidades por meio de mecanismos de segurança.

A aplicação *web* está protegida contra ataques relacionados com os vários tópicos abordados a baixo:

- Git *exposed*: não existem ficheiros públicos sobre o *git*, esta vulnerabilidade quando presente, permite aos invasores acederem ao código-fonte e a vários *branches* diferentes, como *dev* ou produção.
- XSS: não foram encontrados parâmetros nas rotas da aplicação e quando se tenta alterar o URL aparece a mensagem "*{"type": "error", "status":401, "message": "Unauthorized Access"}*". Além disso, foram testados todos os *inputs* aos quais os utilizadores conseguem aceder, mas nenhum deles foi possível tirar proveito da vulnerabilidade XSS.
- SQL *Injection* manual: foram testados todos os *inputs*, mas nenhum está vulneravel a SQL *Injection*.
- *Command Injection*: os vários campos de inserção de dados da aplicação *web* foram testados e mostraram estar seguros em relação à vulnerabilidade de *command injection*.
- CSRF: quando é testada a vulnerabilidade, CSRF os métodos *POST* são rejeitados pelo servidor, transmitindo que a rota não suporta esse tipo de pedidos. Esta vulnerabilidade está corrigida devido à *framework* Laravel.

Também é de notar que as *cookies* utilizam o *SameSite* com o valor de *Lax*, o que ajuda a proteger contra ataques de CSRF.

- *Carriage Return Line Feed* (CRLF): esta vulnerabilidade é reconhecida pela CWE 93 e foi testada na plataforma *web* com a ferramenta Burp Suite[36]. Para este efeito foram realizados vários pedidos e houve a tentativa de modificar a resposta do servidor, mas sem sucesso. *Local File Inclusion*: com esta vulnerabilidade pode ser possível aceder aos ficheiros do servidor. Para testar o *webiste* foram analisados os pedidos que o cliente faz ao servidor, com especial cuidado nos pedidos de imagens. A aplicação *web* retornou sempre o erro de acesso não autorizado.
- *Insecure Direct Object References* (IDOR): não foi encontrado nenhum pedido que devolva um *id* específico, todos os pedidos relacionados com as informações dos utilizadores fazem uma verificação para ver qual é o utilizador autenticado, e como essa informação trata do pedido.
- *Unrestricted file upload*: todos os *inputs* para *upload* de imagens na aplicação, estão restritos a ficheiros do formato *.svg*.
- *Security Missing Configuration*: para avaliar este tipo de vulnerabilidades foi utilizado o método de *guessing* de credenciais na página de *log in* da aplicação e na conexão com o *mySql*. O objetivo foi tentar encontrar alguma credencial padrão de configuração, mas sem sucesso. Houve cuidado em alterar as palavras-passe padrões.
- *Server Side Template Injection* (SSTI): esta vulnerabilidade ocorre quando uma aplicação permite que utilizadores inserirem código de *template* no lado do servidor. Foram executados diversos procedimentos para explorar esta vulnerabilidade, mas tudo o que foi testado não funcionou. A aplicação consegue sempre filtrar como *string*.
- *Insecure deserialization*: não foi encontrado nenhum objeto *serialization*.
- *Brute force*: a página de autenticação está protegida contra esta vulnerabilidade, permite até 5 tentativas. Depois ativa um período de espera de 5 minutos até à rpóxima tentiva. Após 20 tentivas falhadas, a conta é suspensa até o final desse dia. A página de confirmação de identidade para repor a palavra-passe tem o mesmo mecanismo de proteção contra ataques de *brute force*. A página de verificação do código enviado por correio eletrónico permite *brute force*, no entanto, o código só é valido por 10 minutos e tem 12 caracteres, com a possibilidade de ter caracteres especiais. Portanto, não dá para explorar esta vulnerabilidade.

Ao analisar a API *REST* foi detetado que existe potencial para explorar algum tipo de vulnerabilidade na API ou no sistema de sessões. Nesta analise foi utilizada

a ferramenta Postman[96] que facilita na leitura dos pedidos efetuados ao servidor e executados pela API. Rapidamente foi percebido que a aplicação cria um *token* por cada autenticação de utilizador e permite apenas a utilização do *token* mais recente. Também percebeu-se que a aplicação utiliza dois *tokens* distintos para fazer as requisições ao servidor em diferentes momentos.

Existe um *token* inicial responsável pelos pedidos de verificação do utilizador, são pedidos em que o utilizador coloca as suas credenciais corretamente, mas que ainda não confirmou a sua identidade. O outro *token* é o mais utilizado, serve para fazer pedidos e apenas é utilizado para pedidos que exigem que o utilizador esteja autenticado e verificado. E ainda existem os pedidos que não necessitam de nenhum tipo de *token*, por responderem a processos em que nenhum utilizador está autenticado, como efetuar o *log in* ou repor a palavra-passe. Apesar da aplicação parecer muito madura no seu método de autenticação e até incluir um 2FA para verificar a identidade dos utilizadores, é muito fácil de ter algum tipo de vulnerabilidade.

Outro aspeto, que é analisado, são as vulnerabilidades relacionadas com *broken access control* mencionado anteriormente como a vulnerabilidade mais comum pelo OWASP *top* 10. Devido à aplicação ter várias *roles* para os utilizadores, é muito fácil não proteger todas as rotas ou até evitar que um utilizador escale privilégios.

Com auxílio das ferramentas de desenvolvimento do *browser* foram analisados os pedidos realizados por um utilizador *admin* e recriados esses pedidos na ferramenta Postman para explorar pedidos que só determinadas *roles* de utilizadores conseguem executar.

5.5 EXPLOITATION

Ao explorar a aplicação, percebeu-se que quando um utilizador faz a autenticação pela primeira vez é apresentada uma página onde ele pode escolher o 2FA, via endereço de correio eletrónico ou por meio de uma aplicação móvel. Essa página está protegida, contra os utilizadores que configuraram o 2FA para não conseguirem aceder, dá um erro de 403. Porém, é possível fazer o pedido *GET* do QRCode da página onde apenas é possível aceder sem ter configurado o 2FA.

Ao realizar esse pedido ao servidor é retribuído o que é necessário para criar o QRCode, com isto, foi acedido um *website*[97] para criar um QRCode, como fosse a aplicação *web* a realizar este processo. Na Figura 38 é possível observar o novo QRCode.

Com o novo QRCode foi configurado na aplicação Microsoft Authenticator[83] para obter o código de validação. Desta forma, foi possível roubar uma conta de

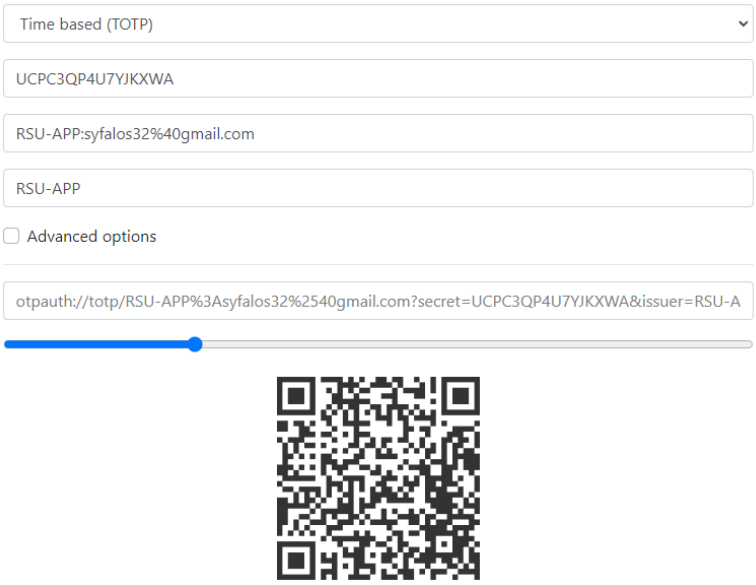


Figura 38: QRCode criado externamente

um utilizador sabendo as credenciais, mas sem ter acesso ao endereço de correio eletrónico do utilizador. O utilizador não consegue passar pelo 2FA e não tem como recuperar, a única solução é repor a palavra-passe antes do invasor e falar com os administradores da ferramenta. Esta vulnerabilidade é considerada média, porque é sempre necessário ter as credenciais do utilizador.

A outra vulnerabilidade foi descoberta devido à analisa realizada aos pedidos efetuados à API. Foram replicados os pedidos que um utilizador *admin* consegue efetuar, em vários utilizadores com diferentes *roles*.

Essa análise levou à descoberta de alguns pedidos que não validam que tipo de utilizador efetua o pedido. Esta vulnerabilidade é considerada crítica e deve ser corrigida.

5.6 *POST-EXPLOITATION*

Nesta etapa do PenTest o objetivo é conseguir manter o acesso e realizar ações adicionais após a exploração. Devido à vulnerabilidade relacionada com *broken access control* é possível ter um utilizador sem privilégios a realizar pedidos que deveriam requerer administração, mas isso não garante que esteja desprotegido durante muito mais tempo.

Para garantir o acesso a um utilizador comum foi realizada uma tentativa de elevar os privilégios do mesmo. Na Figura 39 está representado o pedido efetuado para alterar a *role* do utilizador "worker1" de *worker* para *admin*.

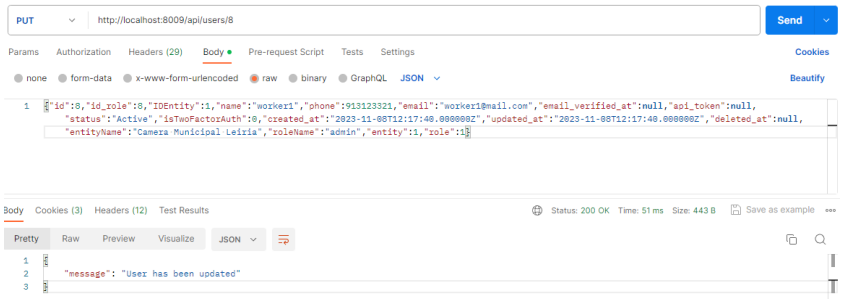


Figura 39: Alterar *role* de um utilizador

Com esta alteração o utilizador *worker1* tem todos os acessos como um administrador normal. Pode alterar tudo o que existe na aplicação, inclusive eliminar os restantes utilizadores, retirando o acesso a todos os outros utilizadores. Caso que a aplicação *web* não tenha um *backup* isto poderia causar o fim de um negócio.

5.7 REPORTING

Visto que foi descrito todo o processo do PenTest ao longo deste capítulo, não existe necessidade de realizar um relatório de PenTest. Não iria acrescentar benefícios, num contexto normal é o relatório do PenTest que é apresentado à identidade cliente que requerer o serviço.

Para além de todo o processo estar descrito, os relatórios por norma também têm uma secção onde são descritas sugestões de melhoria para corrigir todas as vulnerabilidades encontradas. Neste caso, não é necessário porque estas vulnerabilidades são corrigidas antes do trabalho estar concluído.

Apesar de não terem sido encontradas muitas vulnerabilidades, as que foram encontradas são graves e críticas para o funcionamento da aplicação *web*. Isto prova a importância de realizar um PenTest nas aplicações. É muito normal as aplicações que utilizamos diariamente tenham vulnerabilidades como foram descritas, esta aplicação é apenas um exemplo da realidade.

5.8 SÍNTESE

Neste capitulo, foi descrito o PenTest e os testes executados, simulando diversos cenários de ataque. Os testes realizados foram descritos ao longo do capítulo. Durante o processo inclui não apenas a identificação de falhas, mas também sugestões de correções e aprimoramentos para garantir a robustez contínua da aplicação.

Nos capítulos subsequentes, são introduzidas as considerações finais e perspetivas futuras deste trabalho. O capítulo 6 oferece a síntese crítica de todas as descobertas e desafios enfrentados ao longo do desenvolvimento. Posteriormente, no capítulo 7 são delineados as possíveis extensões e aprimoramentos que podem ser explorados para evoluir este projeto.

6

CONCLUSÕES

Este projeto foi delineado com objetivos claros: desenvolver uma plataforma inovadora capaz de gerir sinais rodoviários e RSU, proporcionando a implementação eficaz desses elementos no mundo real. Ao alcançar este propósito, a plataforma não apenas atendeu, mas superou as expectativas, incorporando funcionalidades complementares e fortalecendo os requisitos do sistema. Paralelamente, a ênfase na segurança permeou cada etapa, culminando num processo abrangente de PenTest que não apenas identificou, mas também mitigou vulnerabilidades, conferindo à aplicação um elevado grau de maturidade e robustez.

O desenvolvimento desta plataforma representa um marco significativo no contexto dos ITS, destacando a necessidade premente de aplicações seguras e da realização de PenTest para assegurar a confiabilidade do ecossistema digital. Este trabalho contribui não apenas com uma solução prática, mas também ressalta a importância de abordagens seguras em iniciativas futuras no domínio do ITS.

Ao longo do processo, foram enfrentados vários desafios, desde requisitos complexos e direcionados às necessidades da aplicação, até a complexidade da estrutura das mensagens IVIM que exigiu a criação de 197 tabelas na base de dados. O desenvolvimento seguro implicou a superação de diversas vulnerabilidades, evidenciando a necessidade constante de medidas protetivas. O PenTest, por sua vez, revelou-se uma experiência enriquecedora, não apenas na identificação de fragilidades, mas na implementação efetiva de medidas corretivas.

Do percurso desta jornada, desde a conceção até a implementação e teste, destacam-se os resultados tangíveis e o conhecimento obtido ao enfrentar desafios complexos e em constante evolução.

Por fim, este trabalho não é apenas um ponto de chegada, mas é um ponto de partida para futuras explorações e inovações no domínio vital ITS. A jornada continua, e as lições aprendidas contribuirão para o avanço constante na procura por soluções seguras e eficazes para os desafios do transporte rodoviário na era digital.

Expresso profundo agradecimento aos colegas cuja dedicação e especialização foram fundamentais para o sucesso deste projeto. À equipa responsável que criou os sinais rodoviários físicos e àqueles que implementaram a comunicação entre o conteúdo criado pela aplicação *web* e os sinais, o meu sincero reconhecimento pelo

comprometimento e colaboração excecionais. Graças ao trabalho em conjunto das diversas equipas, foi possível ver este projeto em ação e foi extraordinário.

7

TRABALHO FUTURO

7.1 FUNCIONALIDADES DA APLICAÇÃO

A aplicação permite ter diversos utilizadores com várias permissões diferentes. Seria interessante os administradores da aplicação terem a possibilidade de atribuir permissões individuais, sem ser por roles. Para criar um maior dinamismo nas responsabilidades de cada utilizador e desta forma garantir que as autorizações atribuídas a um determinado utilizador seja suficiente para o mesmo efetuar todas as tarefas a que lhe são atribuídas.

Outra funcionalidade relevante seria uma forma de comunicação entre os utilizadores que são *workers* e que os gestores ou *planners*. Isto porque os *workers* têm de colocar os sinais em terreno com base no que está configurado na aplicação.

Quando é criado um grupo de sinais para implementar numa determinada área e atribuído a diversos *workers*, seria relevante ter um espaço para troca de mensagens, entre os utilizadores para esclarecimento de dúvidas ou para impossibilidades de execução.

Apesar da aplicação estar bem estruturada e os utilizadores conseguirem saber quais são os sinais que estão por implementar em terreno, era importante existir uma forma de distinguir os sinais diretamente no mapa. Desta forma, os utilizadores teriam uma forma de distinguir os sinais que existem em terreno e os que continuam por implementar.

7.2 REFORÇO DE SEGURANÇA NA APLICAÇÃO

Existem várias medidas para tornar a aplicação *web* mais robusta e segura. As organizações que têm várias aplicações e serviços, optam por adquirir um SIEM que permite armazenar os *logs* gerados pela aplicação e posteriormente realizar análises mais complexas num único local.

A realização de pesquisas detalhadas nos *logs* proporciona uma visão abrangente das atividades do sistema, permitindo a identificação proativa de ameaças e resposta a incidentes de segurança eficientemente.

O PenTest realizado teve foco apenas na aplicação *web* sem explorar possíveis vulnerabilidades no servidor onde a mesma está hospedada. Isso abrange uma análise dos componentes da infraestrutura, identificando possíveis vulnerabilidades e pontos fracos que podem ser explorados por invasores.

A análise do código-fonte é fundamental para identificar potenciais vulnerabilidades e garantir a conformidade com boas práticas de programação. Ferramentas como o SonarQube[98] ou o Codacy[99] podem ser integradas no processo de desenvolvimento para realizar verificações automáticas, proporcionando uma análise abrangente do código e promovendo a qualidade e segurança do *software*.

A implementação de uma WAF é uma medida eficaz para proteger a aplicação contra ameaças *online*. Uma WAF atua como uma barreira de segurança, filtrando e monitorizando o tráfego entre a aplicação e a *Internet*. Essa camada adicional de proteção ajuda a mitigar ataques comuns, como SQL *Injection* e XSS.

Como referido, o PenTest realizado é categorizado como *grey box*. Para uma análise mais profunda de segurança, é recomendado realizar um PenTest de categorização de *white box*, que envolve uma avaliação detalhada do código-fonte. Isso proporciona *insights* valiosos sobre possíveis falhas de segurança que podem não ser facilmente identificadas em testes de PenTest *grey box*.

A migração para um protocolo *HyperText Transfer Protocol Secure* (HTTPS) é crucial para garantir a segurança das comunicações entre os utilizadores e a aplicação. O uso de SSL/TLS para criptografar os dados em trânsito protege a interseção contra intercetações e manipulação de dados por terceiros maliciosos.

Estas iniciativas visam elevar a aplicação a padrões ainda mais elevados de segurança, confiabilidade e desempenho, garantindo uma experiência segura e eficaz para os utilizadores. A implementação dessas recomendações solidificará a posição da aplicação como uma solução inovadora e segura no cenário de ITS.

BIBLIOGRAFIA

[1] Brigitte Lonc e Pierpaolo Cincilla. «Cooperative ITS security framework: Standards and implementations progress in Europe». Em: Institute of Electrical e Electronics Engineers Inc., jul. de 2016. ISBN: 9781509021857. DOI: 10.1109/WoWMoM.2016.7523576.

[2] Jonathan Petit e Steven E. Shladover. «Potential Cyberattacks on Automated Vehicles». Em: *IEEE Transactions on Intelligent Transportation Systems* 16 (2 2015). ISSN: 15249050. DOI: 10.1109/TITS.2014.2342271.

[3] Lee Hadlington e Kathryn Parsons. «Can Cyberloafing and Internet Addiction Affect Organizational Information Security?» Em: *Cyberpsychology, Behavior, and Social Networking* 20 (9 set. de 2017), pp. 567–571. ISSN: 21522723. DOI: 10.1089/cyber.2017.0239.

[4] Ana Paula Henriques de Gusmão et al. «Cybersecurity risk analysis model using fault tree analysis and fuzzy decision theory». Em: *International Journal of Information Management* 43 (2018). ISSN: 02684012. DOI: 10.1016/j.ijinfomgt.2018.08.008.

[5] *Cooperative intelligent transport systems (C-ITS) Guidelines on the usage of standards*. 2020.

[6] Muhammad Awais Javed, Sherali Zeadally e Elyes Ben Hamida. «Data analytics for Cooperative Intelligent Transport Systems». Em: *Vehicular Communications* 15 (jan. de 2019), pp. 63–72. ISSN: 22142096. DOI: 10.1016/j.vehcom.2018.10.004.

[7] Andreas Festag. *AUTOMOTIVE NETWORKING AND APPLICATIONS Cooperative Intelligent Transport Systems Standards in Europe*. 2014, p. 166. URL: http://www.autonet.

[8] Bhupendra Singh e Ankit Gupta. «Recent trends in intelligent transportation systems: a review». Em: *Journal of Transport Literature* 9 (2 abr. de 2015), pp. 30–34. DOI: 10.1590/2238-1031.jtl.v9n2a6.

[9] ISO/TC 204. *Intelligent transport systems – Architecture framework*. 2020. URL: https://www.iso.org/standard/73951.html.

[10] Roelf-Erik Carsjens et al. *Development of an ITS-G5 Road Side Unit for Intelligent Transportation Systems*. VDE, 2019. ISBN: 9783800749775.

[11] Maximilian Bauder et al. *Opportunities and challenges of cooperative intelligent transportation systems on accident analysis Development of physical models of lithium-ion batteries for more accurate simulation of electrochemical effects and temperature effects outside operating point [LIBERA]. View project Battery electrolytes View project Opportunities and challenges of cooperative intelligent transportation systems on accident analysis*. 2022. URL: https://www.researchgate.net/publication/366324390.

[12] Jordi Casademont, Elena Lopez-Aguilera e Josep Paradells. «Wake-up radio systems for cooperative-intelligent transport systems architecture». Em: Institute of Electrical e Electronics Engineers Inc., ago. de 2019, pp. 358–363. ISBN: 9781728128887. DOI: 10.1109/FiCloud.2019.00059.

[13] R S Sharma. *MASTER API for intelligent traffic systems and telematics applications implemented on a hybrid communication platform*. 2019.

[14] Politechnika Krakowska et al. *MT-ITS 2019 : 6th International Conference on Models and Technologies for Intelligent Transportation Systems : Cracow University of Technology, 5-7 June 2019, Kraków, Poland*. 2019. ISBN: 9781538694848.

[15] Dries Naudts et al. «Vehicular communication management framework: A flexible hybrid connectivity platform for CCAM services». Em: *Future Internet* 13 (3 mar. de 2021). ISSN: 19995903. DOI: 10.3390/fi13030081.

[16] Alfred Daniel et al. «Cooperative Intelligence of Vehicles for Intelligent Transportation Systems (ITS)». Em: *Wireless Personal Communications* 87 (2 mar. de 2016), pp. 461–484. ISSN: 1572834X. DOI: 10.1007/s11277-015-3078-7.

[17] Rosamaria Elisa Barone et al. «Architecture for parking management in smart cities». Em: *IET Intelligent Transport Systems* 8 (5 2014), pp. 445–452. ISSN: 1751956X. DOI: 10.1049/iet-its.2013.0045.

[18] Shahab Sabzi e Laszlo Vajta. «Security and Energy Consumption Considerations of Electric Vehicles Integration in Smart Grids». Em: *U.Porto Journal of Engineering* 9 (1 2023), pp. 134–149. ISSN: 21836493. DOI: 10.24840/2183-6493_009-001_001382.

[19] Ahmed El-Geneidy Graduate Research Assistant, Robert L Bertini e Assistant Professor. *Integrating Geographic Information Systems and Intelligent Transportation Systems to Improve Incident Management and Life Safety. INTEGRATING GEOGRAPHIC INFORMATION SYSTEMS AND INTELLIGENT TRANSPORTATION SYSTEMS TO IMPROVE INCIDENT MANAGEMENT AND LIFE SAFETY*. 2003.

[20] Aulia Syahirah Khalid et al. «Business Intelligence Dashboard for Driver Performance in Fleet Management». Em: Association for Computing Machinery, out. de 2020, pp. 347–351. ISBN: 9781450372947. DOI: 10.1145/3377571.3377642.

[21] Stephan Lapoehn et al. «Concept of controlling the usage of nomadic devices in highly automated vehicles». Em: vol. 9. Institution of Engineering e Technology, ago. de 2015, pp. 599–605. DOI: 10.1049/iet-its.2014.0208.

[22] I. Ashour, M. Zorkany e M. Shiple. «Design and implementation of transportation management system». Em: SciTePress, 2015, pp. 11–18. ISBN: 9789897581090. DOI: 10.5220/0005430200110018.

[23] Fabio Arena e Giovanni Pau. «An overview of vehicular communications». Em: *Future Internet* 11 (2 jan. de 2019). ISSN: 19995903. DOI: 10.3390/fi11020027.

[24] Vivek Katiyar, Prashant Kumar e Narottam Chand. «An Intelligent Transportation Systems Architecture using Wireless Sensor Networks». Em: *International Journal of Computer Applications* 14 (2 jan. de 2011), pp. 22–26. DOI: 10.5120/1816-2369.

[25] Lukas Malina et al. «On Security and Privacy in Vehicle Speed-Limiting Services in the Internet of Vehicles». Em: *IEEE Intelligent Transportation Systems Magazine* (jan. de 2022). ISSN: 19411197. DOI: 10.1109/MITS.2022.3203236.

[26] Steven E. Shladover. «Connected and automated vehicle systems: Introduction and overview». Em: *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations* 22 (3 mai. de 2018), pp. 190–200. ISSN: 15472442. DOI: 10.1080/15472450.2017.1336053.

[27] OWASP. *Homepage do OWASP*. URL: https://owasp.org/Top10/. (accessed: 11.01.2022).

[28] Institute SANS. *Homepage do SANS*. URL: https://www.sans.org/emea/. (accessed: 12.01.2022).

[29] CIS. *Homepage do CIS*. URL: https://www.cisecurity.org/controls. (accessed: 13.01.2022).

[30] CWE. *Homepage do CWE*. URL: https://cwe.mitre.org/index.html. (accessed: 25.01.2022).

[31] OWASP. *OWASP Top 10:2021*. URL: https://owasp.org/Top10/. (accessed: 05.12.2022).

[32] Matthew Bach-Nutman. *Understanding The Top 10 OWASP Vulnerabilities*. 2020.

[33] Synopsys. *What is OWASP TOP 10*. URL: https://www.synopsys.com/glossary/what-is-owasp-top-10.html. (accessed: 06.01.2022).

[34] OWASP. *OWASP Top Ten Project*. https://owasp.org/Top10/. Acessado em 8 de fevereiro de 2023. 2021.

[35] Mahfuz Ibne Hannan Azaz Ahamed Touseef Aziz Khan. «Automated Testing: Testing Top 10 OWASP Vulnerabilities of Government Web Applications in Bangladesh». Em: (2022).

[36] PortSwigger. *Burp Suite*. URL: https://portswigger.net/burp.

[37] The OWASP Foundation. *OWASP Zed Attack Proxy (ZAP)*. URL: https://www.zaproxy.org/.

[38] Invicti Security. *Invicti Security*. URL: https://www.invicti.com/.

[39] Diah Priyawati, Siti Rokhmah e Ihsan Cahyo Utomo. *Website Vulnerability Testing and Analysis of Internet Management Information System Using OWASP*. 2022, pp. 2745–9659. URL: https://ijcis.net/index.php/ijcis/index.

[40] *CWE - CWE-1344: Use of Externally-Controlled Format String*. https://cwe.mitre.org/data/definitions/1344.html. Acessado em 8 de fevereiro de 2023.

[41] CWE. *Weaknesses in OWASP Top 10:2021*. URL: https://cwe.mitre.org/data/definitions/1344.html. (accessed: 10.01.2022).

[42] *CWE - CWE-11: Time-of-check Time-of-use (TOCTOU) Race Condition*. https://cwe.mitre.org/data/definitions/11.html. Acessado em 8 de fevereiro de 2023.

[43] SANS Institute. *SANS CIS controls*. URL: https://www.sans.org/blog/cis-controls-v8/. (accessed: 13.01.2022).

[44] SANS Institute. *The SANS Top 25 Most Dangerous Software Errors*. https://www.sans.org/top25-software-errors/. Acessado em 8 de fevereiro de 2023. 2021.

[45] *CIS Controls Version 8*. https://www.cisecurity.org/controls/v8/. Acessado em 8 de fevereiro de 2023.

[46] Stjepan Gros. «A Critical View on CIS Controls». Em: Institute of Electrical e Electronics Engineers Inc., jun. de 2021, pp. 122–128. ISBN: 9789531842716. DOI: 10.23919/ConTEL52528.2021.9495982.

[47] CIS. *CIS CAT*. URL: https://www.cisecurity.org/cybersecurity-tools/cis-cat-pro.

[48] Philip Kobezak et al. *Host Inventory Controls and Systems Survey: Evaluating the CIS Critical Security Control One in Higher Education Networks*. 2018. URL: http://hdl.handle.net/10125/50486.

[49] RSA. *RSA Conference*. URL: https://www.rsaconference.com/.

[50] The MITRE Corporation. *Common Weakness Enumeration - About*. https://cwe.mitre.org/about/. acesso em 8 de fevereiro de 2023.

[51] Bob Martin et al. *CWE - 2010 CWE/SANS Top 25 Most Dangerous Software Errors*. 2010. URL: http://cwe.mitre.org/top25/.

[52] Vladimir Dimitrov e Ivan Kolev. *An Ontology of Top 25 CWEs*. 2020.

[53] Jinfeng Li. «Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST)». Em: *Annals of Emerging Technologies in Computing* 4 (3 jul. de 2020), pp. 1–8. ISSN: 2516029X. DOI: 10.33166/AETiC.2020.03.001.

[54] Niek Jan van den Hout. *Standardised Penetration Testing? Examining the Usefulness of Current Penetration Testing Methodologies*. 2019. URL: https://www.researchgate.net/publication/335652869.

[55] Abdurrahman Pektas e Ertugrul Basaranoglu. «Practical Approach for Securing Windows Environment : Attack Vectors and Countermeasures». Em: *International Journal of Network Security & Its Applications* 9 (6 nov. de 2017), pp. 13–27. ISSN: 09752307. DOI: 10.5121/ijnsa.2017.9602.

[56] Pete Herzog. *The Open Source Security Testing Methodology Manual*. 2008. URL: https://www.isecom.org/research.html.

[57] OWASP. *OWASP Web Security Testing Guide*. OWASP. 2014. URL: https://owasp.org/www-project-web-security-testing-guide/.

[58] Software Secured. «NIST SP 800-115 and Penetration Testing». Em: (mar. de 2018). URL: https://www.softwaresecured.com/nist-sp-800-115-and-penetration-testing/.

[59] PTES. *Penetration Testing Execution Standard (PTES)*. 2014. URL: http://www.pentest-standard.org/index.php/Main_Page.

[60] FutureLearn. *Ethical Hacking: Information System Security Assessment Framework (ISSAF)*. 2022. URL: https://www.futurelearn.com/info/courses/ethical-hacking-an-introduction/0/steps/71521.

[61] ISECOM. *Open Source Security Testing Methodology Manual (OSSTMM) Version 3*. 2016.

[62] OWASP. *OWASP Web Security Testing Guide (v4.2)*. 2021.

[63] OWASP. *OWASP Testing Guide v4.2 - Penetration Testing Methodologies*. https://owasp.org/www-project-web-security-testing-guide/v41/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies.

[64] K A Scarfone et al. *Technical guide to information security testing and assessment*. National Institute of Standards e Technology, 2008. DOI: 10.6028/NIST.SP.800-115. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf.

[65] Penetration Testing Execution Standard (PTES). *PTES Technical Guidelines*. URL: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines.

[66] EC-Council. *Improve Penetration Testing ROI*. https://www.eccouncil.org/cybersecurity-exchange/penetration-testing/improve-penetration-testing-roi/.

[67] Sapphire. *Penetration Testing Methodology*. https://www.sapphire.net/cybersecurity/penetration-testing-methodology/.

[68] Jahanzeb Shahid et al. «A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions». Em: *Applied Sciences (Switzerland)* 12 (8 abr. de 2022). ISSN: 20763417. DOI: 10.3390/app12084077.

[69] Nuno Teodoro. *Web application security: Improving critical web-based applications quality through in-depth security analysis*. 2011. ISBN: 978095642638/3.

[70] Eurium Technologies. *OWASP Top 10 (2021) Threat Levels & Examples*. URL: https://www.euriun.com/owasp/.

[71] MITRE Corporation. *CWE Top 25 Most Dangerous Software Weaknesses – 2022*. 2022. URL: https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html.

[72] The Hacker News. *Does OWASP Top 10 Still Matter?* 2022. URL: https://thehackernews.com/2022/10/does-owasp-top-10-still-matter.html.

[73] Parth Sane. «Is the OWASP Top 10 List Comprehensive Enough for Writing Secure Code?» Em: Association for Computing Machinery, mai. de 2020, pp. 58–61. ISBN: 9781450375061. DOI: 10.1145/3437075.3437089.

[74] OWASP. *OWASP Top 10 A01:2021 – Broken Access Control*. 2021. URL: https://owasp.org/Top10/A01_2021-Broken_Access_Control/.

[75] OWASP. *OWASP Top 10 A02:2021 – Cryptographic Failures*. 2021. URL: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/.

[76] OWASP. *OWASP Top 10 A03:2021 – Injection*. 2021. URL: https://owasp.org/Top10/A03_2021-Injection/.

[77] OWASP. *ModSecurity*. URL: https://owasp.org/www-project-modsecurity-core-rule-set/.

[78] Elasticsearch. *Elasticsearch*. URL: https://www.elastic.co/pt/.

[79] Elasticsearch. *logstash*. URL: https://www.elastic.co/pt/logstash.

[80] Elasticsearch. *kibana*. URL: https://www.elastic.co/pt/kibana.

[81] Layer 8 Security. *Penetration Testing - PTES Methodology*. URL: https://layer8security.com/services/penetration-testing/.

[82] Google Corporation. *Google Authenticator*. URL: https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&pli=1.

[83] Microsoft Corporation. *Microsoft Authenticator*. URL: https://play.google.com/store/apps/details?id=com.azure.authenticator.

[84] packagist. *packagist*. URL: https://packagist.org/.

[85] *Regular Expressions 101*. URL: https://regex101.com/.

[86] Telnet. *Telnet*. URL: https://www.telnet.org/.

[87] GNU. *wget*. URL: https://www.gnu.org/software/wget/.

[88] Joona Hoikkala. *ffuf*. URL: https://github.com/ffuf/ffuf.

[89] danielmiessler. *SecLists*. URL: https://github.com/danielmiessler/SecLists.

[90] Aboul3la. *Sublist3r*. URL: https://github.com/aboul3la/Sublist3r.

[91] Edge-Security. *wfuzz*. URL: https://www.edge-security.com/wfuzz.php.

[92] dirb. *dirb*. URL: https://dirb.sourceforge.net/.

[93] Nmap. *Namp*. URL: https://nmap.org/.

[94] wappalyzer. *wappalyzer*. URL: https://www.wappalyzer.com/.

[95] Morning Star Security. *whatweb*. URL: https://morningstarsecurity.com/research/whatweb.

[96] Postman. *Postman*. URL: https://www.postman.com/.

[97] Stefan Sundin. *2FA QR Code Generator*. URL: https://stefansundin.github.io/2fa-qr/.

[98] Sonar. *Sonar Qube*. URL: https://www.sonarsource.com/products/sonarqube/.

[99] Codacy. *Codacy*. URL: https://www.codacy.com/.

APÊNDICE A

Neste apêndice, é apresentado o diagrama da base de dados desenvolvida para a aplicação *web*. A construção da base de dados teve como fundamento a estrutura da mensagem IVIM. A Figura 40 representa o diagrama da base de dados.

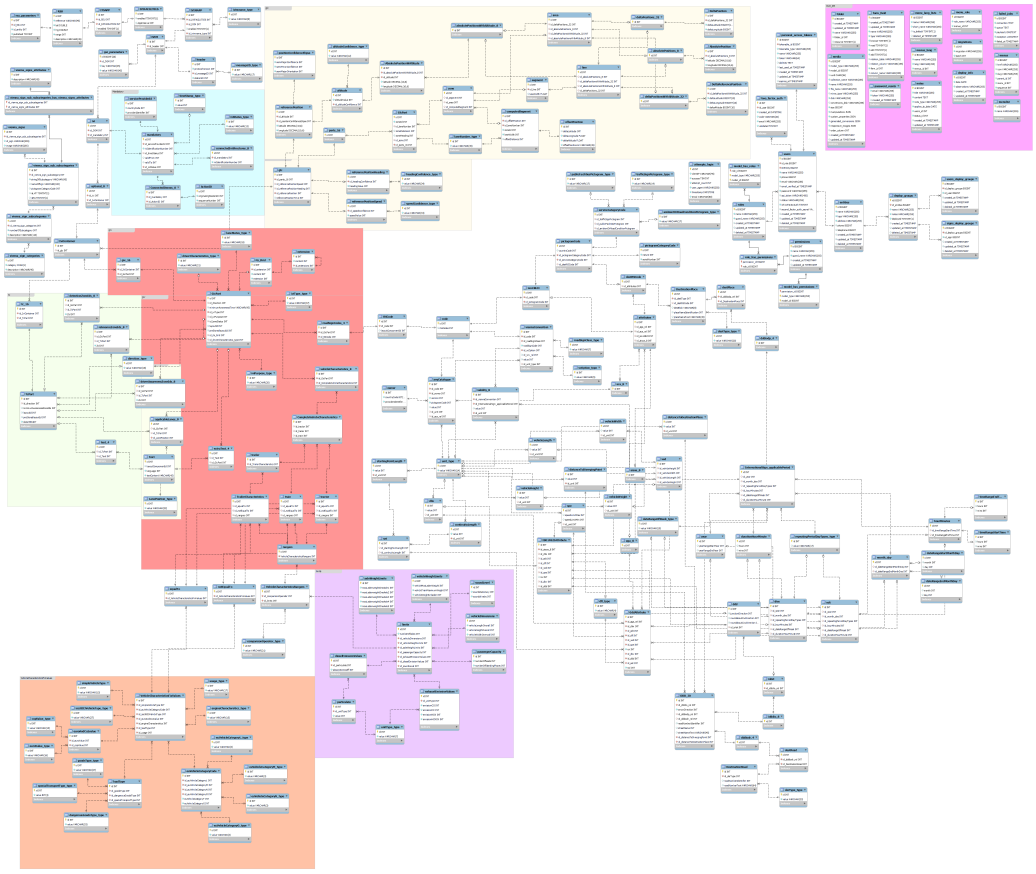


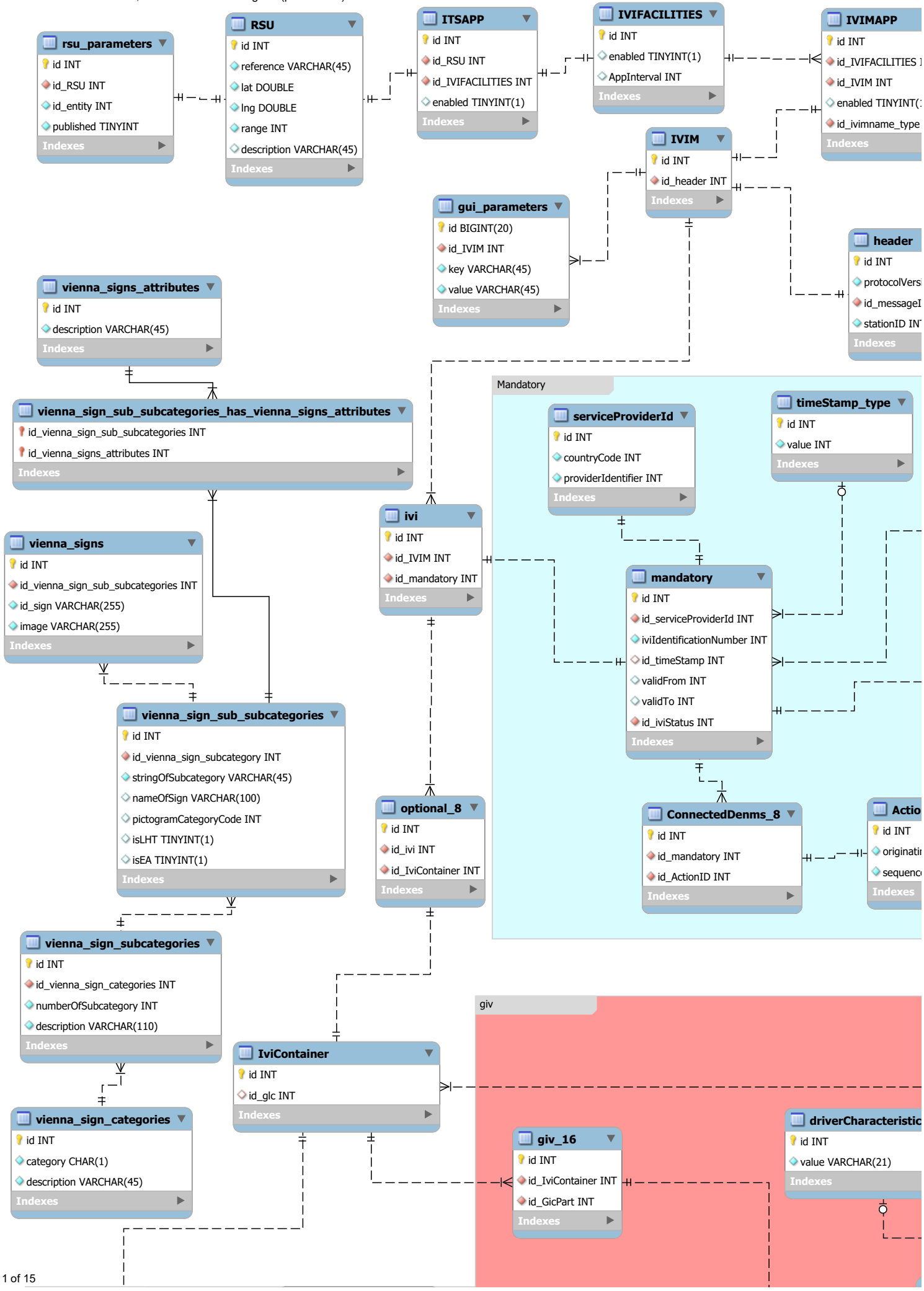
Figura 40: Diagrama da base de dados

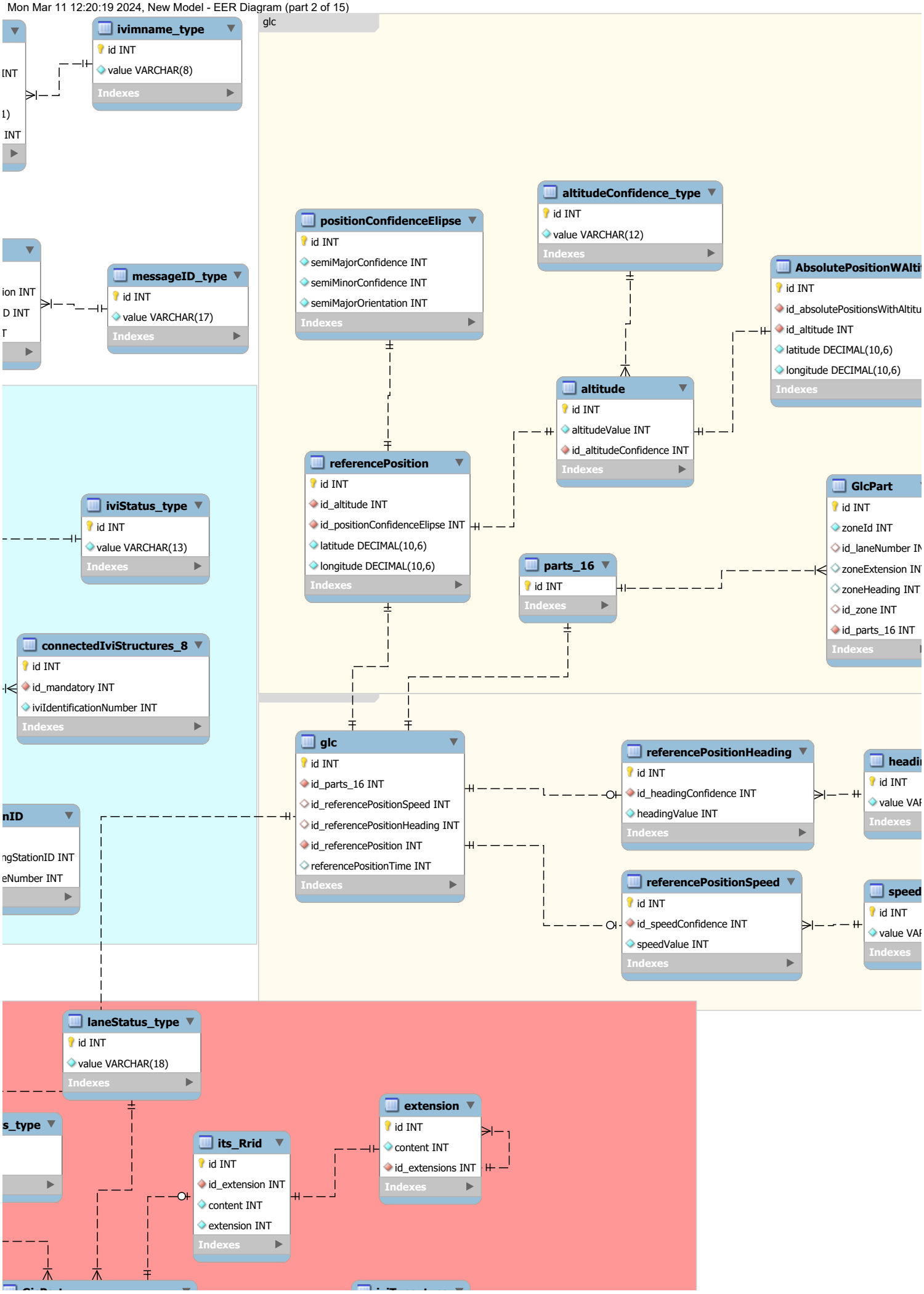
Como é possível observar, trata-se de uma base de dados bastante complexa, principalmente devido ao seu número elevado de tabelas, contendo 197 tabelas. Para facilitar de alguma forma a divisão das tabelas na base de dados, foram criadas áreas coloridas no diagrama:

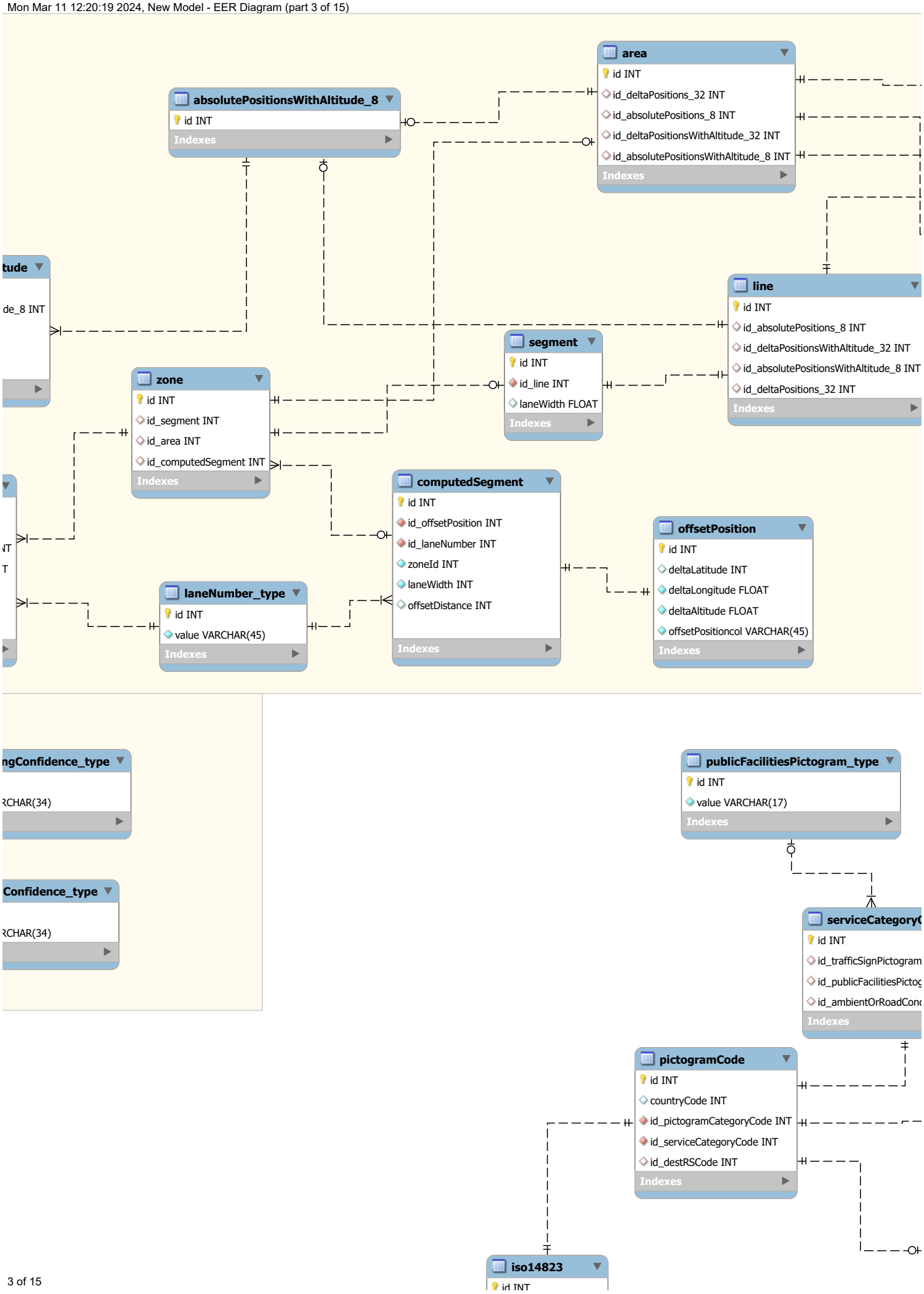
- Azul: tabelas relacionados com o *mandatory* da mensagem IVIM.
- Vermelho: tabelas associadas ao GIC.
- Verde: tabelas do TC.
- Amarelo: tabelas relacionadas com o GLC

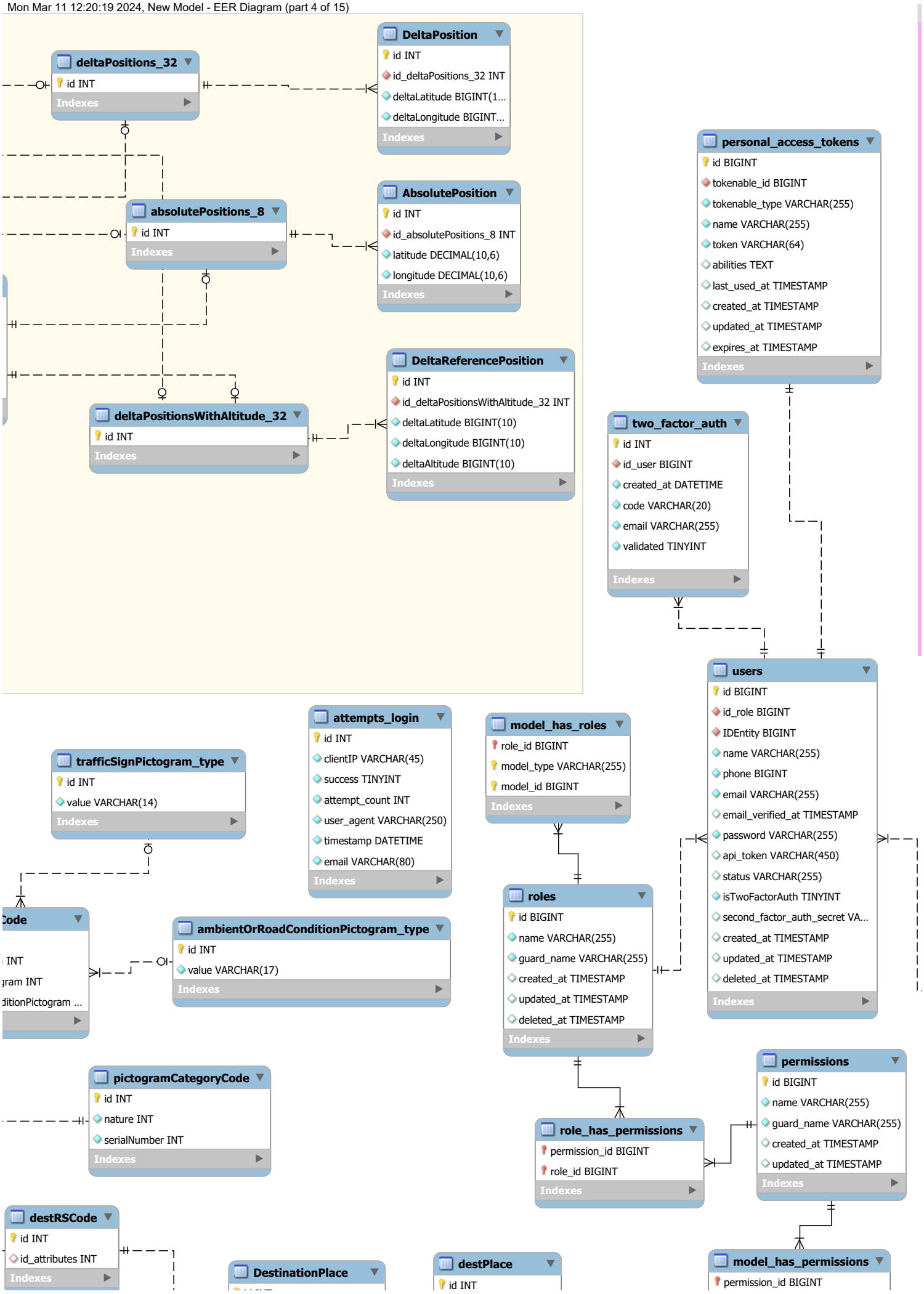
- Laranja: tabelas associadas a valores fixos das características dos veículos.
- Roxo: tabelas de ligação dos diversos *containers*, sobre limites relacionados com os veículos (dimensões, pesos, cargas, entre outros).
- Rosa: tabelas da primeira base de dados que não foram eliminadas por razões históricas.

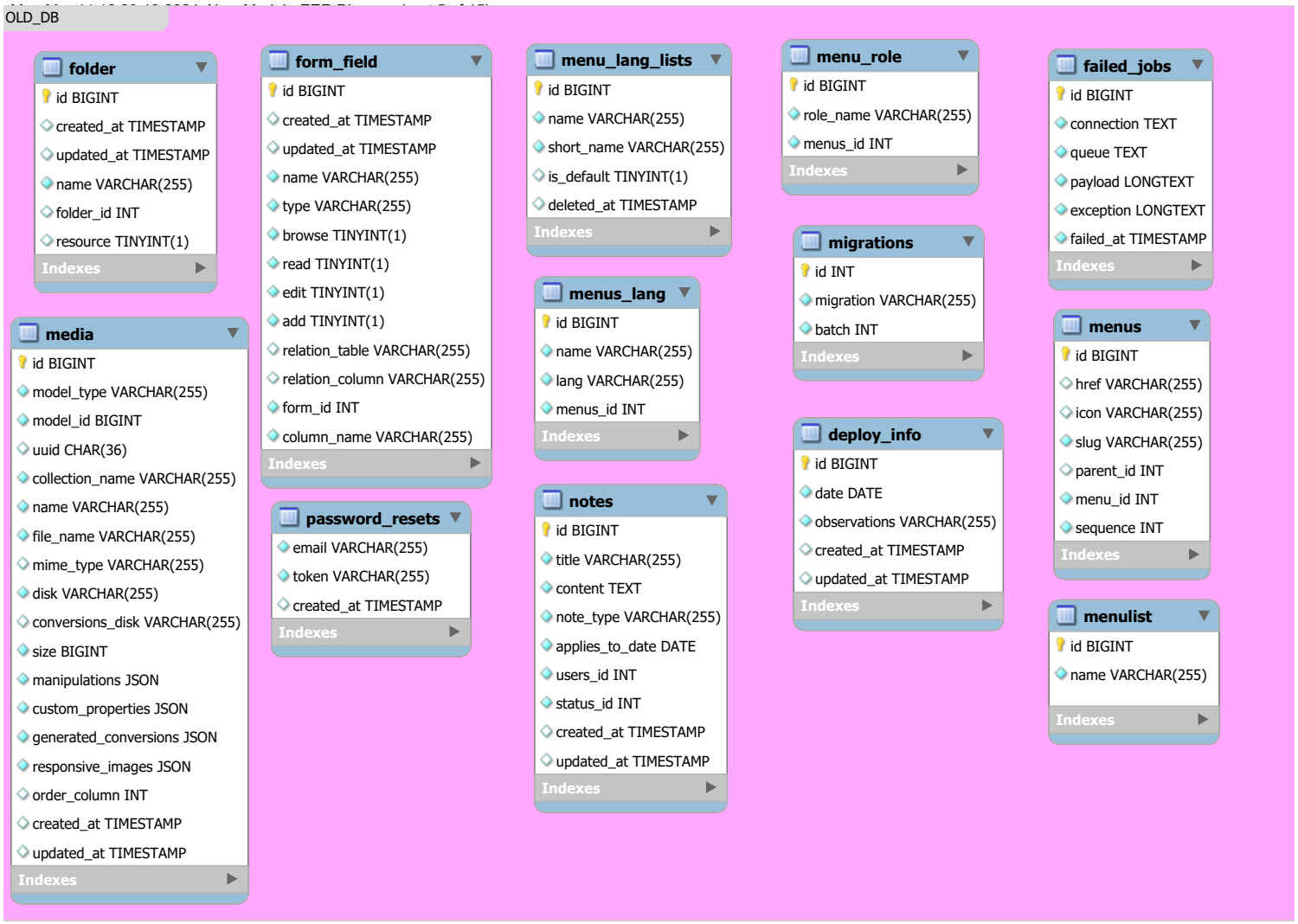
Nas próximas 15 páginas são apresentadas as tabelas da base de dados amplamente para ser legível ao leitor.

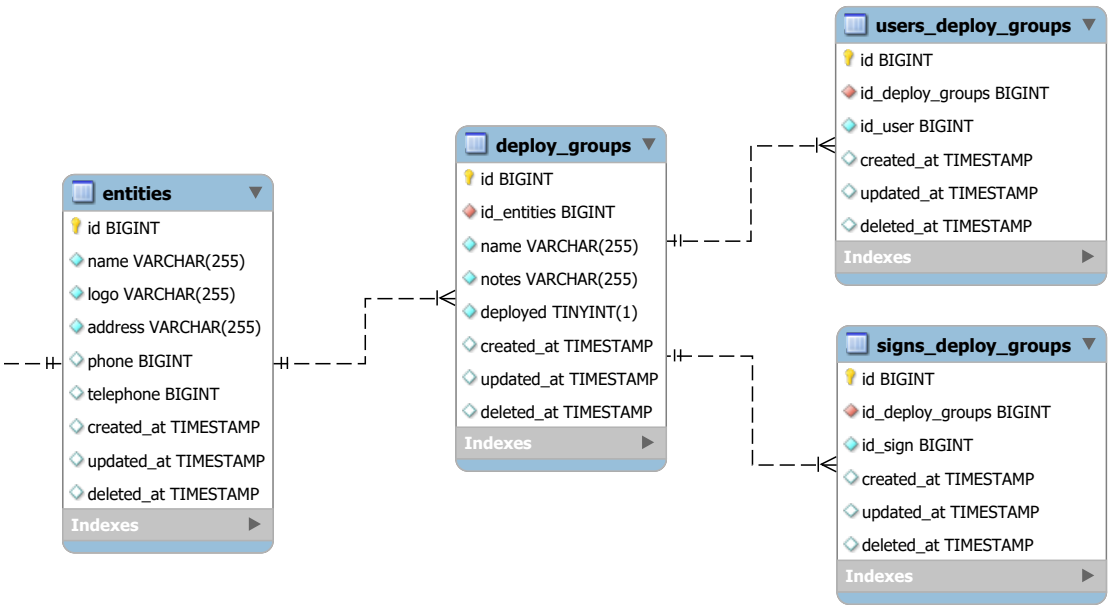

rsu_parameters
id INT
id_RSU INT
id_entity INT
published TINYINT
Indexes
RSU
id INT
reference VARCHAR(45)
lat DOUBLE
lng DOUBLE
range INT
description VARCHAR(45)
Indexes
ITSAPP
id INT
id_RSU INT
id_IVIFACILITIES INT
enabled TINYINT(1)
Indexes
IVIFACILITIES
id INT
enabled TINYINT(1)
AppInterval INT
Indexes
IVIMAPP
id INT
id_IVIFACILITIES
id_IVIM INT
enabled TINYINT(
id_ivimname_type
Indexes
IVIM
id INT
id_header INT
Indexes
gui_parameters
id BIGINT(20)
id_IVIM INT
key VARCHAR(45)
value VARCHAR(45)
Indexes
header
id INT
protocolVers
id_messageI
stationID INT
Indexes
vienna_signs_attributes
id INT
description VARCHAR(45)
Indexes
vienna_sign_sub_subcategories_has_vienna_signs_attributes
id_vienna_sign_sub_subcategories INT
id_vienna_signs_attributes INT
Indexes
Mandatory
serviceProviderId
id INT
countryCode INT
providerIdentifier INT
Indexes
timeStamp_type
id INT
value INT
Indexes
ivi
id INT
id_IVIM INT
id_mandatory INT
Indexes
vienna_signs
id INT
id_vienna_sign_sub_subcategories INT
id_sign VARCHAR(255)
image VARCHAR(255)
Indexes
mandatory
id INT
id_serviceProviderId INT
iviIdentificationNumber INT
id_timeStamp INT
validFrom INT
validTo INT
id_iviStatus INT
Indexes
vienna_sign_sub_subcategories
id INT
id_vienna_sign_subcategory INT
stringOfSubcategory VARCHAR(45)
nameOfSign VARCHAR(100)
pictogramCategoryCode INT
isLHT TINYINT(1)
isEA TINYINT(1)
Indexes
optional_8
id INT
id_ivi INT
id_IviContainer INT
Indexes
ConnectedDenms_8
id INT
id_mandatory INT
id_ActionID INT
Indexes
Actio
id INT
originati
sequenc
Indexes
vienna_sign_subcategories
id INT
id_vienna_sign_categories INT
numberOfSubcategory INT
description VARCHAR(110)
Indexes
giv
IviContainer
id INT
id_glc INT
Indexes
vienna_sign_categories
id INT
category CHAR(1)
description VARCHAR(45)
Indexes
giv_16
id INT
id_IviContainer INT
id_GicPart INT
Indexes
driverCharacteristic
id INT
value VARCHAR(21)
Indexes

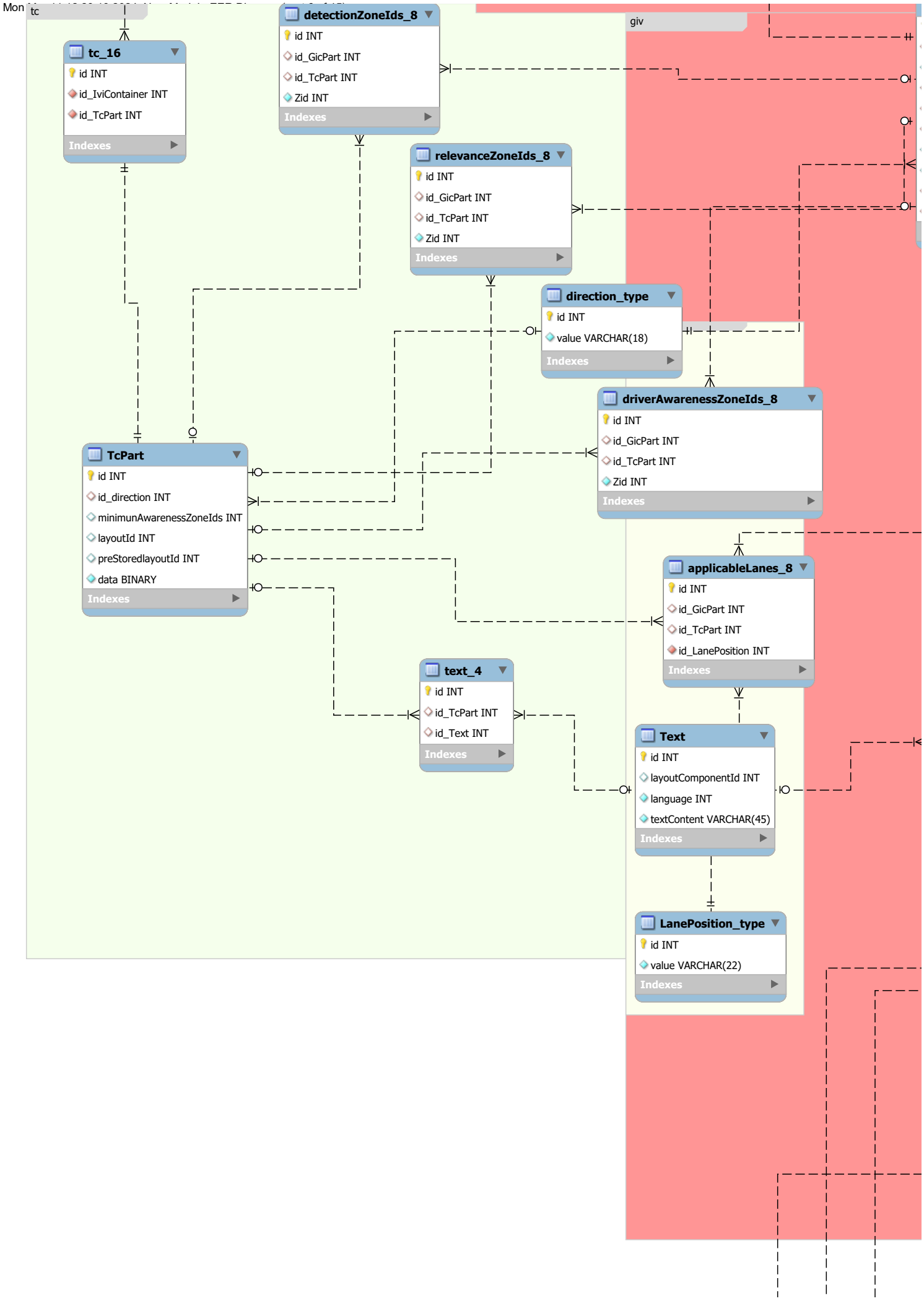

glc
ivimname_type
id INT
value VARCHAR(8)
Indexes
messageID_type
id INT
value VARCHAR(17)
Indexes
positionConfidenceElipse
id INT
semiMajorConfidence INT
semiMinorConfidence INT
semiMajorOrientation INT
Indexes
altitudeConfidence_type
id INT
value VARCHAR(12)
Indexes
AbsolutePositionWAlti
id INT
id_absolutePositionsWithAltitu
id_altitude INT
latitude DECIMAL(10,6)
longitude DECIMAL(10,6)
Indexes
altitude
id INT
altitudeValue INT
id_altitudeConfidence INT
Indexes
referencePosition
id INT
id_altitude INT
id_positionConfidenceElipse INT
latitude DECIMAL(10,6)
longitude DECIMAL(10,6)
Indexes
iviStatus_type
id INT
value VARCHAR(13)
Indexes
parts_16
id INT
Indexes
GlcPart
id INT
zoneId INT
id_laneNumber IN
zoneExtension IN
zoneHeading INT
id_zone INT
id_parts_16 INT
Indexes
connectedIviStructures_8
id INT
id_mandatory INT
iviIdentificationNumber INT
Indexes
glc
id INT
id_parts_16 INT
id_referencePositionSpeed INT
id_referencePositionHeading INT
id_referencePosition INT
referencePositionTime INT
Indexes
referencePositionHeading
id INT
id_headingConfidence INT
headingValue INT
Indexes
referencePositionSpeed
id INT
id_speedConfidence INT
speedValue INT
Indexes
laneStatus_type
id INT
value VARCHAR(18)
Indexes
its_Rrid
id INT
id_extension INT
content INT
extension INT
Indexes
extension
id INT
content INT
id_extensions INT
Indexes

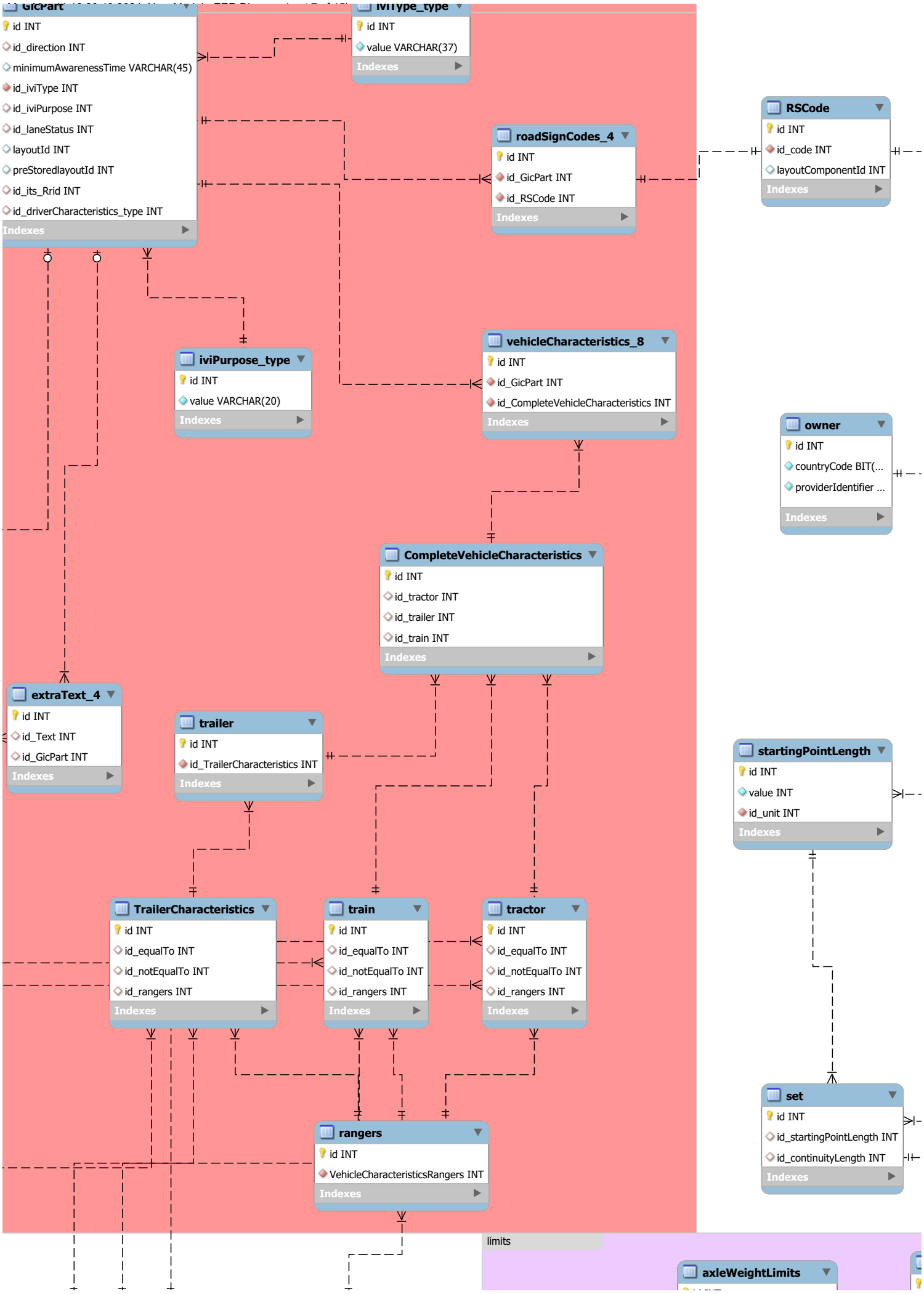

area
id INT
id_deltaPositions_32 INT
id_absolutePositions_8 INT
id_deltaPositionsWithAltitude_32 INT
id_absolutePositionsWithAltitude_8 INT
Indexes
absolutePositionsWithAltitude_8
id INT
Indexes
line
id INT
id_absolutePositions_8 INT
id_deltaPositionsWithAltitude_32 INT
id_absolutePositionsWithAltitude_8 INT
id_deltaPositions_32 INT
Indexes
segment
id INT
id_line INT
laneWidth FLOAT
Indexes
zone
id INT
id_segment INT
id_area INT
id_computedSegment INT
Indexes
computedSegment
id INT
id_offsetPosition INT
id_laneNumber INT
zoneId INT
laneWidth INT
offsetDistance INT
Indexes
offsetPosition
id INT
deltaLatitude INT
deltaLongitude FLOAT
deltaAltitude FLOAT
offsetPositioncol VARCHAR(45)
Indexes
laneNumber_type
id INT
value VARCHAR(45)
Indexes
publicFacilitiesPictogram_type
id INT
value VARCHAR(17)
Indexes
pictogramCode
id INT
countryCode INT
id_pictogramCategoryCode INT
id_serviceCategoryCode INT
id_destRSCode INT
Indexes
iso14823
id INT

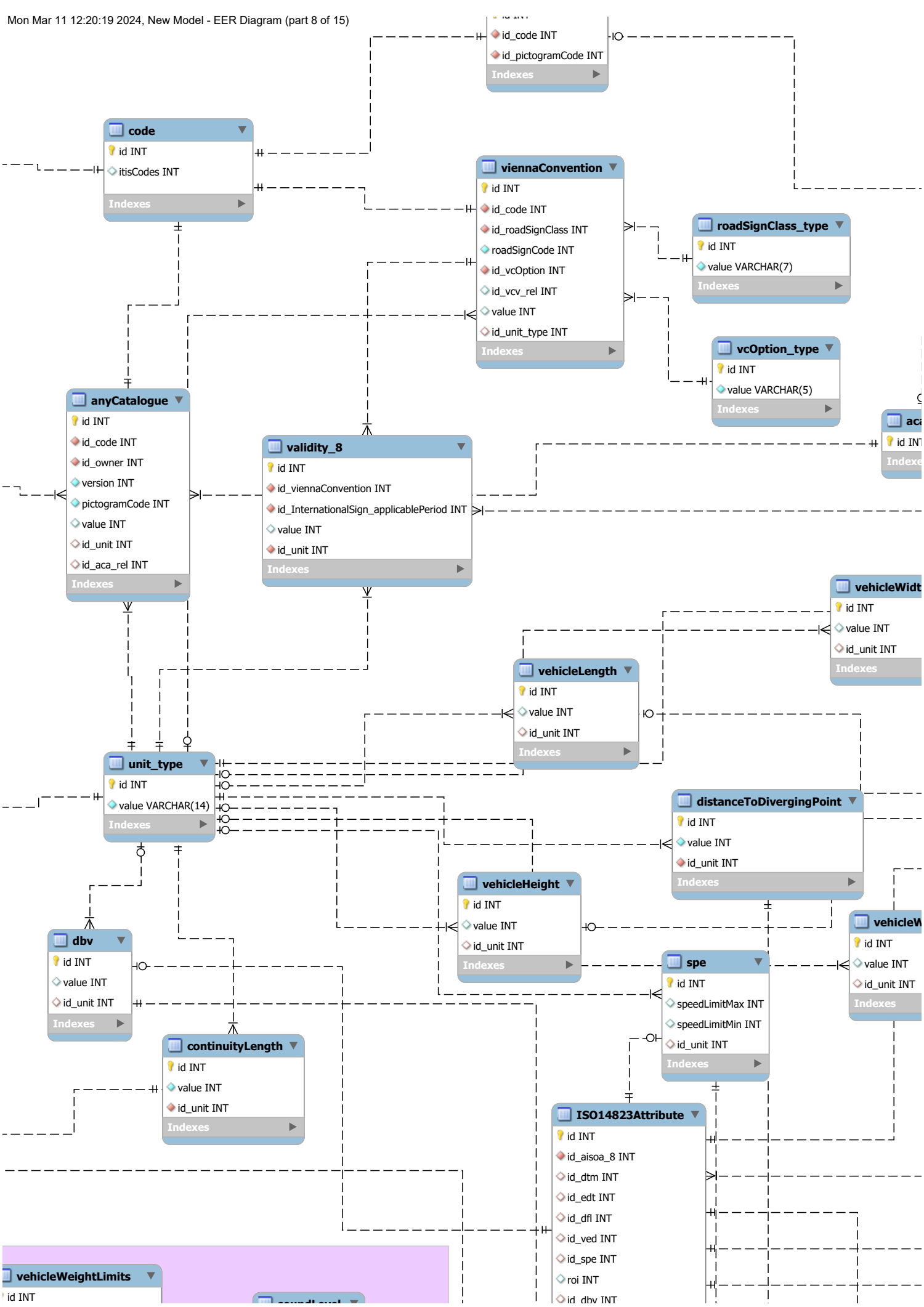

deltaPositions_32
id INT
Indexes
DeltaPosition
id INT
id_deltaPositions_32 INT
deltaLatitude BIGINT(1...
deltaLongitude BIGINT...
Indexes
absolutePositions_8
id INT
Indexes
AbsolutePosition
id INT
id_absolutePositions_8 INT
latitude DECIMAL(10,6)
longitude DECIMAL(10,6)
Indexes
deltaPositionsWithAltitude_32
id INT
Indexes
DeltaReferencePosition
id INT
id_deltaPositionsWithAltitude_32 INT
deltaLatitude BIGINT(10)
deltaLongitude BIGINT(10)
deltaAltitude BIGINT(10)
Indexes
personal_access_tokens
id BIGINT
tokenable_id BIGINT
tokenable_type VARCHAR(255)
name VARCHAR(255)
token VARCHAR(64)
abilities TEXT
last_used_at TIMESTAMP
created_at TIMESTAMP
updated_at TIMESTAMP
expires_at TIMESTAMP
Indexes
two_factor_auth
id INT
id_user BIGINT
created_at DATETIME
code VARCHAR(20)
email VARCHAR(255)
validated TINYINT
Indexes
users
id BIGINT
id_role BIGINT
IDEntity BIGINT
name VARCHAR(255)
phone BIGINT
email VARCHAR(255)
email_verified_at TIMESTAMP
password VARCHAR(255)
api_token VARCHAR(450)
status VARCHAR(255)
isTwoFactorAuth TINYINT
second_factor_auth_secret VA...
created_at TIMESTAMP
updated_at TIMESTAMP
deleted_at TIMESTAMP
Indexes
attempts_login
id INT
clientIP VARCHAR(45)
success TINYINT
attempt_count INT
user_agent VARCHAR(250)
timestamp DATETIME
email VARCHAR(80)
Indexes
model_has_roles
role_id BIGINT
model_type VARCHAR(255)
model_id BIGINT
Indexes
trafficSignPictogram_type
id INT
value VARCHAR(14)
Indexes
roles
id BIGINT
name VARCHAR(255)
guard_name VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
deleted_at TIMESTAMP
Indexes
ambientOrRoadConditionPictogram_type
id INT
value VARCHAR(17)
Indexes
permissions
id BIGINT
name VARCHAR(255)
guard_name VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
role_has_permissions
permission_id BIGINT
role_id BIGINT
Indexes
pictogramCategoryCode
id INT
nature INT
serialNumber INT
Indexes
destRSCode
id INT
id_attributes INT
Indexes
DestinationPlace
destPlace
id INT
model_has_permissions
permission_id BIGINT

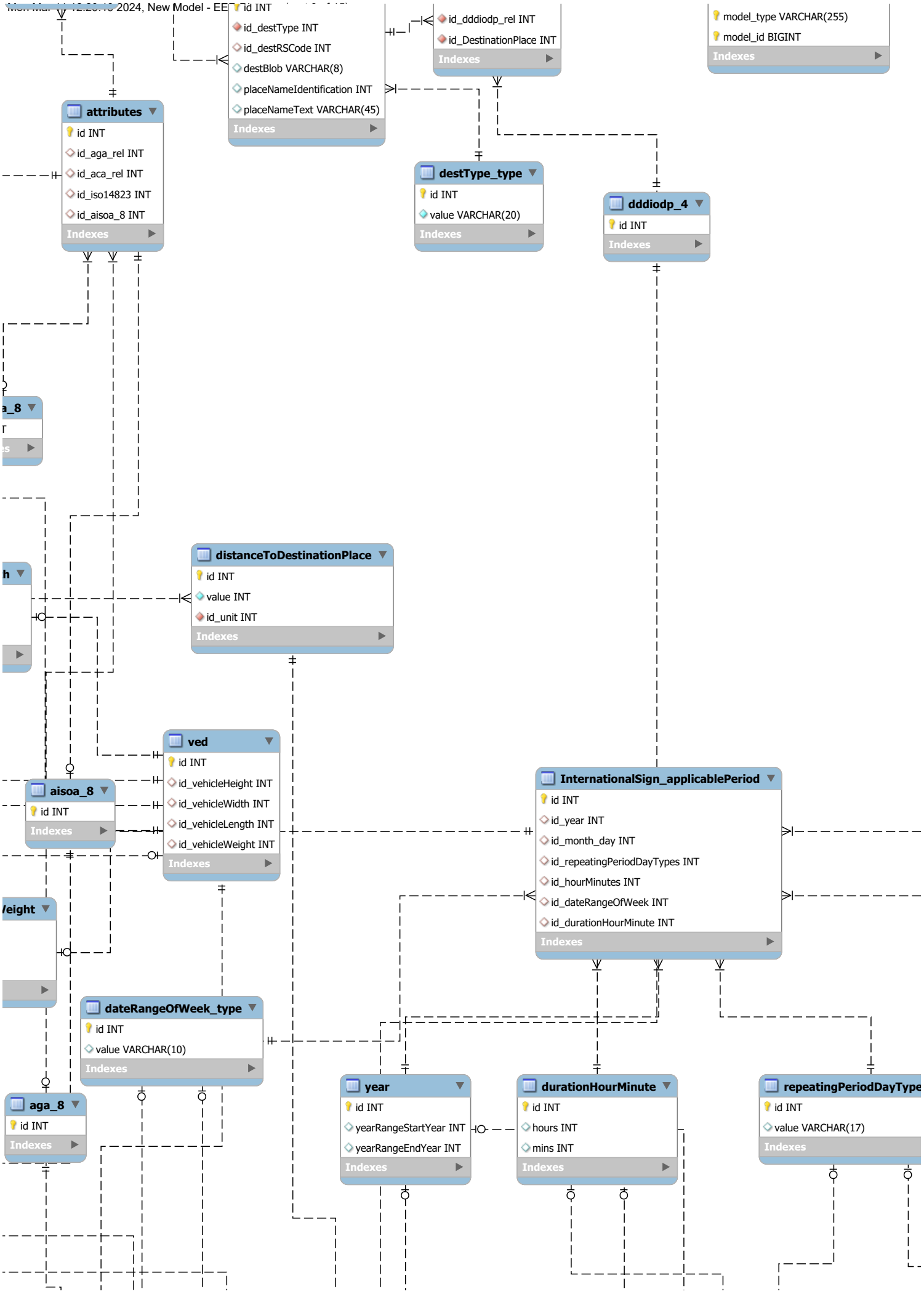

OLD_DB
folder
id BIGINT
created_at TIMESTAMP
updated_at TIMESTAMP
name VARCHAR(255)
folder_id INT
resource TINYINT(1)
Indexes
form_field
id BIGINT
created_at TIMESTAMP
updated_at TIMESTAMP
name VARCHAR(255)
type VARCHAR(255)
browse TINYINT(1)
read TINYINT(1)
edit TINYINT(1)
add TINYINT(1)
relation_table VARCHAR(255)
relation_column VARCHAR(255)
form_id INT
column_name VARCHAR(255)
Indexes
menu_lang_lists
id BIGINT
name VARCHAR(255)
short_name VARCHAR(255)
is_default TINYINT(1)
deleted_at TIMESTAMP
Indexes
menu_role
id BIGINT
role_name VARCHAR(255)
menus_id INT
Indexes
failed_jobs
id BIGINT
connection TEXT
queue TEXT
payload LONGTEXT
exception LONGTEXT
failed_at TIMESTAMP
Indexes
media
id BIGINT
model_type VARCHAR(255)
model_id BIGINT
uuid CHAR(36)
collection_name VARCHAR(255)
name VARCHAR(255)
file_name VARCHAR(255)
mime_type VARCHAR(255)
disk VARCHAR(255)
conversions_disk VARCHAR(255)
size BIGINT
manipulations JSON
custom_properties JSON
generated_conversions JSON
responsive_images JSON
order_column INT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
menus_lang
id BIGINT
name VARCHAR(255)
lang VARCHAR(255)
menus_id INT
Indexes
migrations
id INT
migration VARCHAR(255)
batch INT
Indexes
menus
id BIGINT
href VARCHAR(255)
icon VARCHAR(255)
slug VARCHAR(255)
parent_id INT
menu_id INT
sequence INT
Indexes
password_resets
email VARCHAR(255)
token VARCHAR(255)
created_at TIMESTAMP
Indexes
notes
id BIGINT
title VARCHAR(255)
content TEXT
note_type VARCHAR(255)
applies_to_date DATE
users_id INT
status_id INT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
deploy_info
id BIGINT
date DATE
observations VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
menulist
id BIGINT
name VARCHAR(255)
Indexes

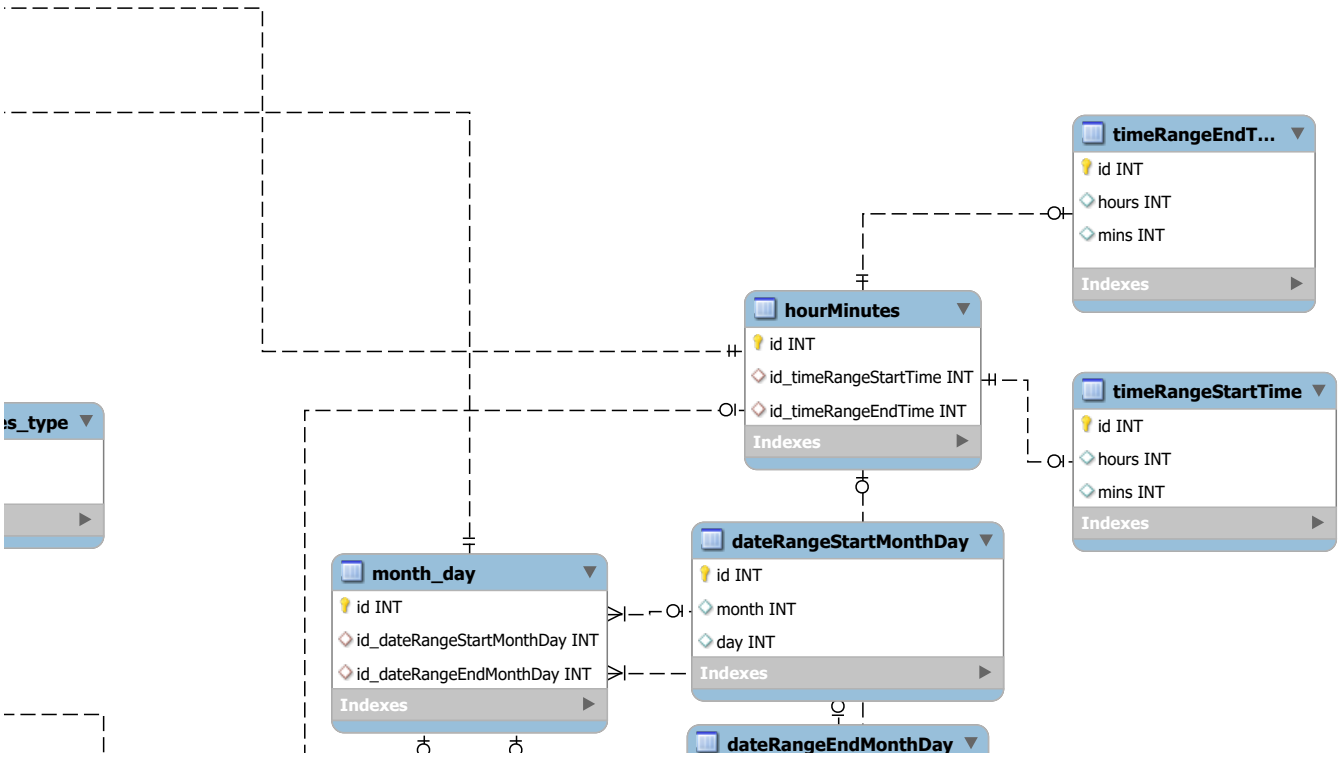

users_deploy_groups
id BIGINT
id_deploy_groups BIGINT
id_user BIGINT
created_at TIMESTAMP
updated_at TIMESTAMP
deleted_at TIMESTAMP
Indexes
deploy_groups
id BIGINT
id_entities BIGINT
name VARCHAR(255)
notes VARCHAR(255)
deployed TINYINT(1)
created_at TIMESTAMP
updated_at TIMESTAMP
deleted_at TIMESTAMP
Indexes
entities
id BIGINT
name VARCHAR(255)
logo VARCHAR(255)
address VARCHAR(255)
phone BIGINT
telephone BIGINT
created_at TIMESTAMP
updated_at TIMESTAMP
deleted_at TIMESTAMP
Indexes
signs_deploy_groups
id BIGINT
id_deploy_groups BIGINT
id_sign BIGINT
created_at TIMESTAMP
updated_at TIMESTAMP
deleted_at TIMESTAMP
Indexes

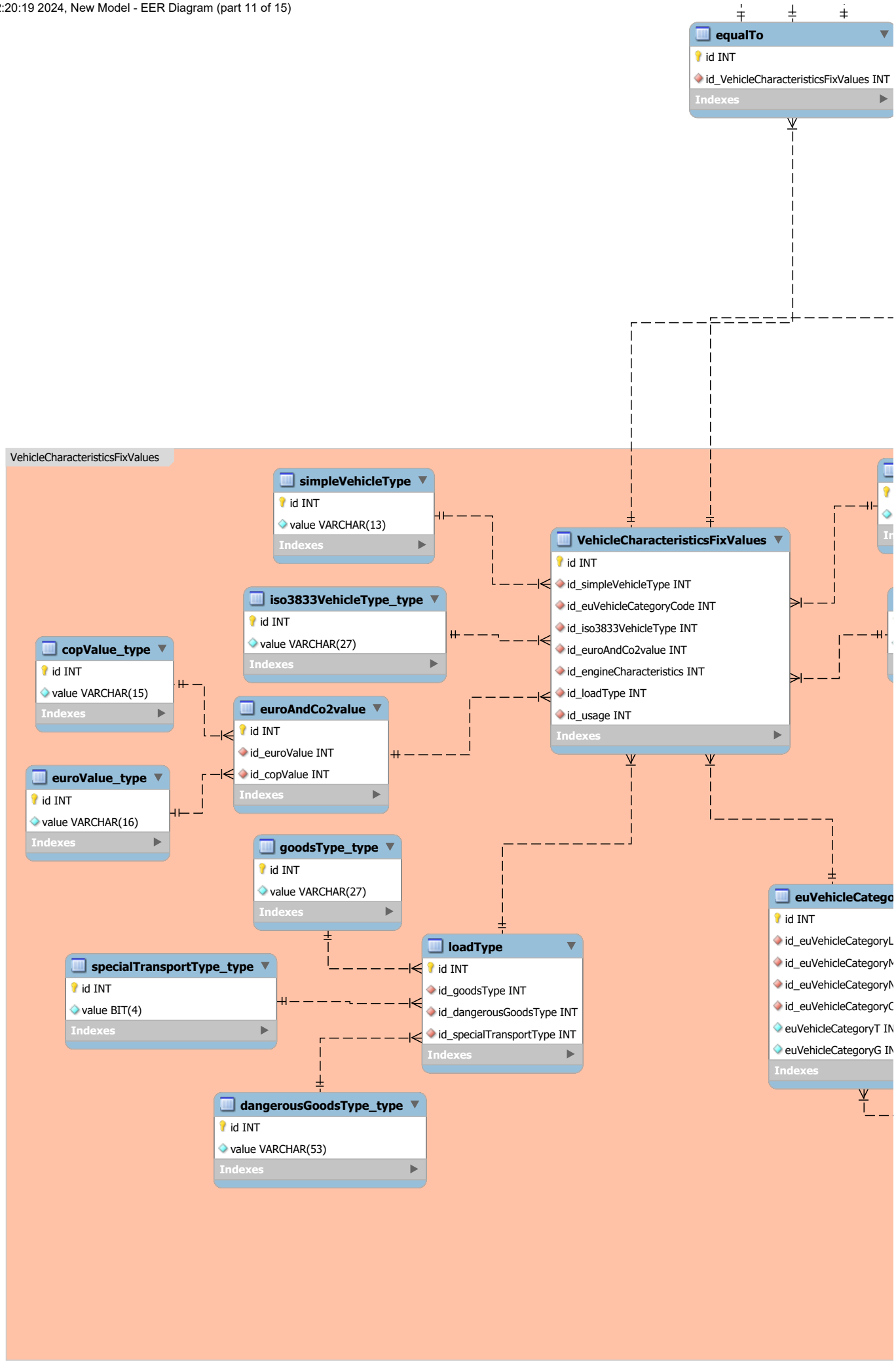

tc
giv
tc_16
id INT
id_IviContainer INT
id_TcPart INT
Indexes
detectionZoneIds_8
id INT
id_GicPart INT
id_TcPart INT
Zid INT
Indexes
relevanceZoneIds_8
id INT
id_GicPart INT
id_TcPart INT
Zid INT
Indexes
direction_type
id INT
value VARCHAR(18)
Indexes
driverAwarenessZoneIds_8
id INT
id_GicPart INT
id_TcPart INT
Zid INT
Indexes
TcPart
id INT
id_direction INT
minimunAwarenessZoneIds INT
layoutId INT
preStoredlayoutId INT
data BINARY
Indexes
applicableLanes_8
id INT
id_GicPart INT
id_TcPart INT
id_LanePosition INT
Indexes
text_4
id INT
id_TcPart INT
id_Text INT
Indexes
Text
id INT
layoutComponentId INT
language INT
textContent VARCHAR(45)
Indexes
LanePosition_type
id INT
value VARCHAR(22)
Indexes

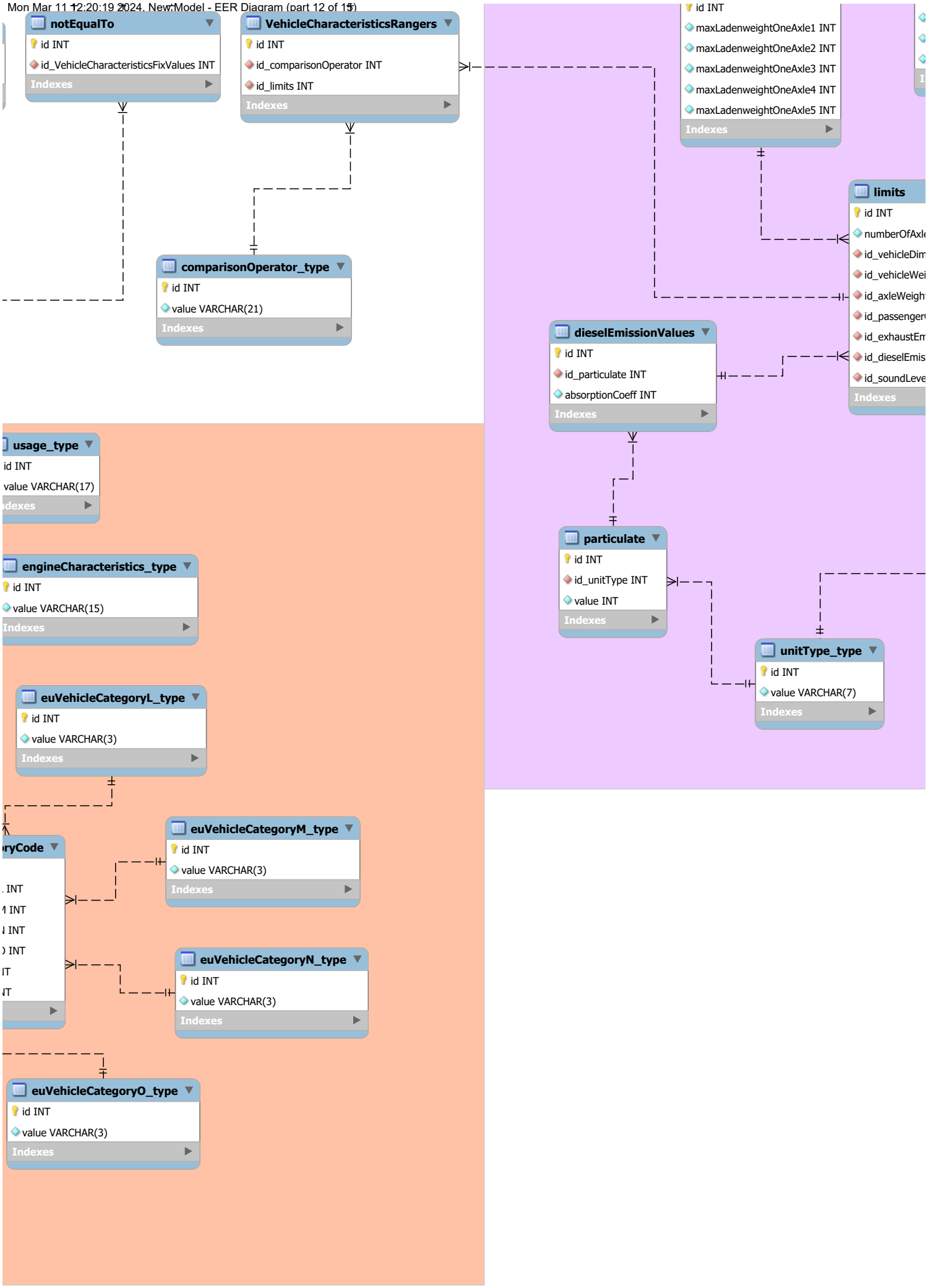

GicPart
id INT
id_direction INT
minimumAwarenessTime VARCHAR(45)
id_iviType INT
id_iviPurpose INT
id_laneStatus INT
layoutId INT
preStoredlayoutId INT
id_its_Rrid INT
id_driverCharacteristics_type INT
Indexes
iviType_type
id INT
value VARCHAR(37)
Indexes
RSCode
id INT
id_code INT
layoutComponentId INT
Indexes
roadSignCodes_4
id INT
id_GicPart INT
id_RSCode INT
Indexes
iviPurpose_type
id INT
value VARCHAR(20)
Indexes
vehicleCharacteristics_8
id INT
id_GicPart INT
id_CompleteVehicleCharacteristics INT
Indexes
owner
id INT
countryCode BIT(...
providerIdentifier ...
Indexes
CompleteVehicleCharacteristics
id INT
id_tractor INT
id_trailer INT
id_train INT
Indexes
extraText_4
id INT
id_Text INT
id_GicPart INT
Indexes
trailer
id INT
id_TrailerCharacteristics INT
Indexes
startingPointLength
id INT
value INT
id_unit INT
Indexes
TrailerCharacteristics
id INT
id_equalTo INT
id_notEqualTo INT
id_rangers INT
Indexes
train
id INT
id_equalTo INT
id_notEqualTo INT
id_rangers INT
Indexes
tractor
id INT
id_equalTo INT
id_notEqualTo INT
id_rangers INT
Indexes
set
id INT
id_startingPointLength INT
id_continuityLength INT
Indexes
rangers
id INT
VehicleCharacteristicsRangers INT
Indexes
limits
axleWeightLimits

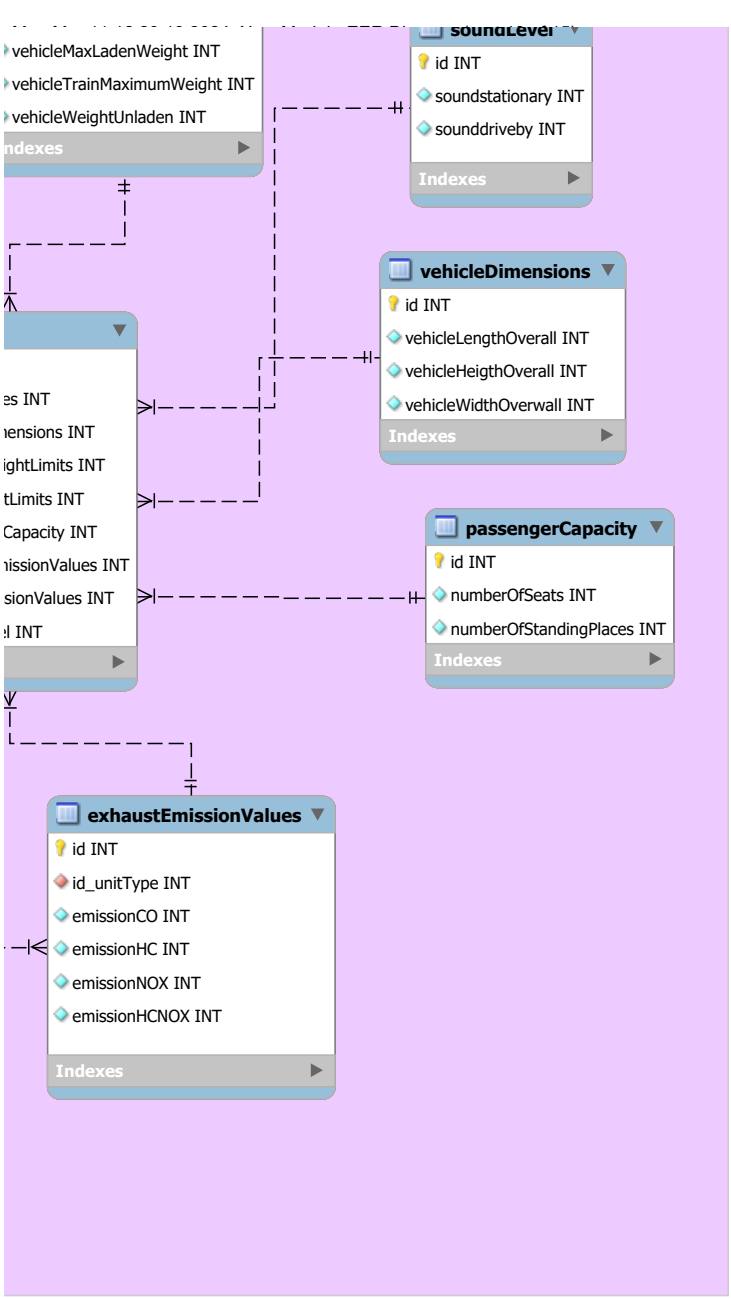

id_code INT
id_pictogramCode INT
Indexes
code
id INT
itisCodes INT
Indexes
viennaConvention
id INT
id_code INT
id_roadSignClass INT
roadSignCode INT
id_vcOption INT
id_vcv_rel INT
value INT
id_unit_type INT
Indexes
roadSignClass_type
id INT
value VARCHAR(7)
Indexes
vcOption_type
id INT
value VARCHAR(5)
Indexes
anyCatalogue
id INT
id_code INT
id_owner INT
version INT
pictogramCode INT
value INT
id_unit INT
id_aca_rel INT
Indexes
validity_8
id INT
id_viennaConvention INT
id_InternationalSign_applicablePeriod INT
value INT
id_unit INT
Indexes
vehicleWidt
id INT
value INT
id_unit INT
Indexes
vehicleLength
id INT
value INT
id_unit INT
Indexes
unit_type
id INT
value VARCHAR(14)
Indexes
distanceToDivergingPoint
id INT
value INT
id_unit INT
Indexes
vehicleHeight
id INT
value INT
id_unit INT
Indexes
dbv
id INT
value INT
id_unit INT
Indexes
spe
id INT
speedLimitMax INT
speedLimitMin INT
id_unit INT
Indexes
vehicleW
id INT
value INT
id_unit INT
Indexes
continuityLength
id INT
value INT
id_unit INT
Indexes
ISO14823Attribute
id INT
id_aisoa_8 INT
id_dtm INT
id_edt INT
id_dfl INT
id_ved INT
id_spe INT
roi INT
id_dbv INT
vehicleWeightLimits
id INT

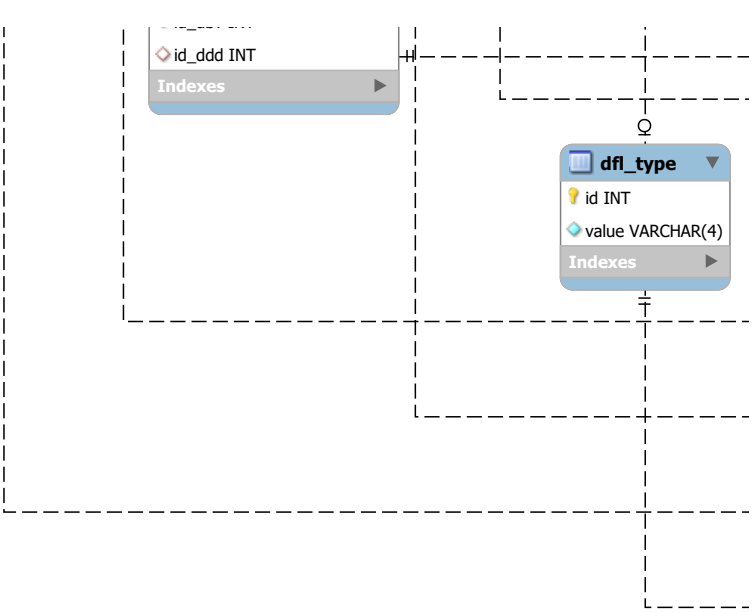

2024, New Model - EE
id INT
id_destType INT
id_destRSCode INT
destBlob VARCHAR(8)
placeNameIdentification INT
placeNameText VARCHAR(45)
Indexes
id_dddiodp_rel INT
id_DestinationPlace INT
Indexes
model_type VARCHAR(255)
model_id BIGINT
Indexes
attributes
id INT
id_aga_rel INT
id_aca_rel INT
id_iso14823 INT
id_aisoa_8 INT
Indexes
destType_type
id INT
value VARCHAR(20)
Indexes
dddiodp_4
id INT
Indexes
distanceToDestinationPlace
id INT
value INT
id_unit INT
Indexes
ved
id INT
id_vehicleHeight INT
id_vehicleWidth INT
id_vehicleLength INT
id_vehicleWeight INT
Indexes
aisoa_8
id INT
Indexes
InternationalSign_applicablePeriod
id INT
id_year INT
id_month_day INT
id_repeatingPeriodDayTypes INT
id_hourMinutes INT
id_dateRangeOfWeek INT
id_durationHourMinute INT
Indexes
dateRangeOfWeek_type
id INT
value VARCHAR(10)
Indexes
year
id INT
yearRangeStartYear INT
yearRangeEndYear INT
Indexes
durationHourMinute
id INT
hours INT
mins INT
Indexes
repeatingPeriodDayType
id INT
value VARCHAR(17)
Indexes
aga_8
id INT
Indexes

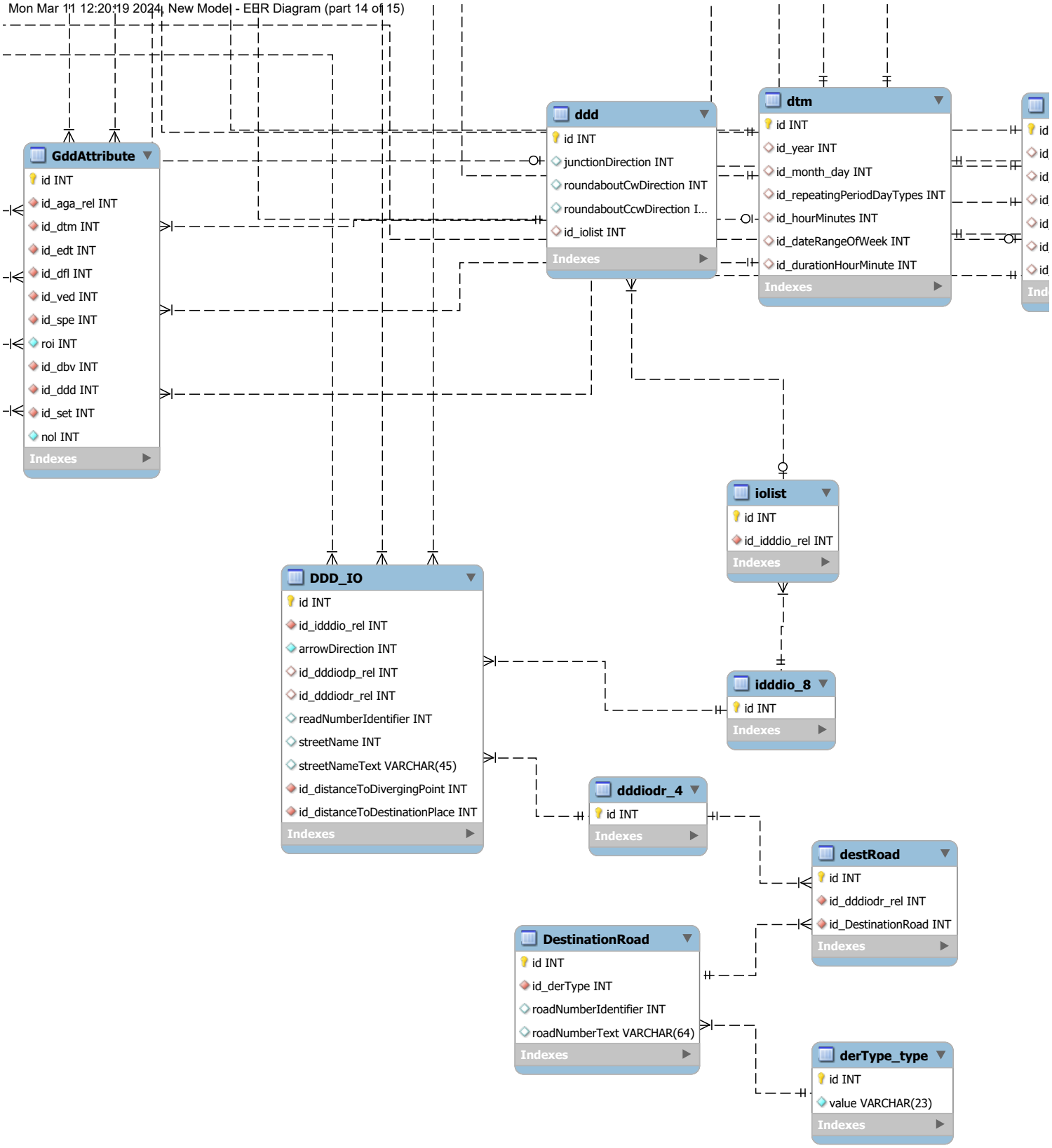

timeRangeEndT...
id INT
hours INT
mins INT
Indexes
hourMinutes
id INT
id_timeRangeStartTime INT
id_timeRangeEndTime INT
Indexes
timeRangeStartTime
id INT
hours INT
mins INT
Indexes
dateRangeStartMonthDay
id INT
month INT
day INT
Indexes
month_day
id INT
id_dateRangeStartMonthDay INT
id_dateRangeEndMonthDay INT
Indexes
dateRangeEndMonthDay

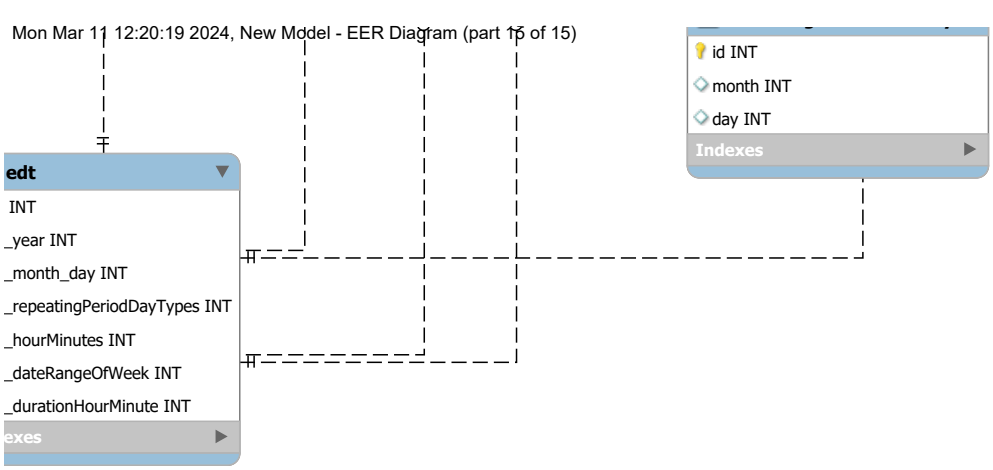

equalTo
id INT
id_VehicleCharacteristicsFixValues INT
Indexes
VehicleCharacteristicsFixValues
simpleVehicleType
id INT
value VARCHAR(13)
Indexes
VehicleCharacteristicsFixValues
id INT
id_simpleVehicleType INT
id_euVehicleCategoryCode INT
id_iso3833VehicleType INT
id_euroAndCo2value INT
id_engineCharacteristics INT
id_loadType INT
id_usage INT
Indexes
iso3833VehicleType_type
id INT
value VARCHAR(27)
Indexes
copValue_type
id INT
value VARCHAR(15)
Indexes
euroAndCo2value
id INT
id_euroValue INT
id_copValue INT
Indexes
euroValue_type
id INT
value VARCHAR(16)
Indexes
goodsType_type
id INT
value VARCHAR(27)
Indexes
loadType
id INT
id_goodsType INT
id_dangerousGoodsType INT
id_specialTransportType INT
Indexes
specialTransportType_type
id INT
value BIT(4)
Indexes
dangerousGoodsType_type
id INT
value VARCHAR(53)
Indexes
euVehicleCatego
id INT
Indexes


notEqualTo
id INT
id_VehicleCharacteristicsFixValues INT
Indexes
VehicleCharacteristicsRangers
id INT
id_comparisonOperator INT
id_limits INT
Indexes
comparisonOperator_type
id INT
value VARCHAR(21)
Indexes
id INT
maxLadenweightOneAxle1 INT
maxLadenweightOneAxle2 INT
maxLadenweightOneAxle3 INT
maxLadenweightOneAxle4 INT
maxLadenweightOneAxle5 INT
Indexes
limits
id INT
numberOfAxl
id_vehicleDim
id_vehicleWei
id_axleWeigh
id_passenger
id_exhaustEm
id_dieselEmis
id_soundLeve
Indexes
dieselEmissionValues
id INT
id_particulate INT
absorptionCoeff INT
Indexes
usage_type
id INT
value VARCHAR(17)
Indexes
particulate
id INT
id_unitType INT
value INT
Indexes
unitType_type
id INT
value VARCHAR(7)
Indexes
engineCharacteristics_type
id INT
value VARCHAR(15)
Indexes
euVehicleCategoryL_type
id INT
value VARCHAR(3)
Indexes
euVehicleCategoryM_type
id INT
value VARCHAR(3)
Indexes
euVehicleCategoryN_type
id INT
value VARCHAR(3)
Indexes
euVehicleCategoryO_type
id INT
value VARCHAR(3)
Indexes


vehicleMaxLadenWeight INT
vehicleTrainMaximumWeight INT
vehicleWeightUnladen INT
Indexes
soundLevel
id INT
soundstationary INT
sounddriveby INT
Indexes
vehicleDimensions
id INT
vehicleLengthOverall INT
vehicleHeigthOverall INT
vehicleWidthOverwall INT
Indexes
passengerCapacity
id INT
numberOfSeats INT
numberOfStandingPlaces INT
Indexes
exhaustEmissionValues
id INT
id_unitType INT
emissionCO INT
emissionHC INT
emissionNOX INT
emissionHCNOX INT
Indexes


id_ddd INT
Indexes
dfl_type
id INT
value VARCHAR(4)
Indexes


GddAttribute
id INT
id_aga_rel INT
id_dtm INT
id_edt INT
id_dfl INT
id_ved INT
id_spe INT
roi INT
id_dbv INT
id_ddd INT
id_set INT
nol INT
Indexes
ddd
id INT
junctionDirection INT
roundaboutCwDirection INT
roundaboutCcwDirection I...
id_iolist INT
Indexes
dtm
id INT
id_year INT
id_month_day INT
id_repeatingPeriodDayTypes INT
id_hourMinutes INT
id_dateRangeOfWeek INT
id_durationHourMinute INT
Indexes
iolist
id INT
id_idddio_rel INT
Indexes
DDD_IO
id INT
id_idddio_rel INT
arrowDirection INT
id_dddiodp_rel INT
id_dddiodr_rel INT
readNumberIdentifier INT
streetName INT
streetNameText VARCHAR(45)
id_distanceToDivergingPoint INT
id_distanceToDestinationPlace INT
Indexes
idddio_8
id INT
Indexes
dddiodr_4
id INT
Indexes
destRoad
id INT
id_dddiodr_rel INT
id_DestinationRoad INT
Indexes
DestinationRoad
id INT
id_derType INT
roadNumberIdentifier INT
roadNumberText VARCHAR(64)
Indexes
derType_type
id INT
value VARCHAR(23)
Indexes


id INT
month INT
day INT
Indexes
edt
INT
_year INT
_month_day INT
_repeatingPeriodDayTypes INT
_hourMinutes INT
_dateRangeOfWeek INT
_durationHourMinute INT
exes

DECLARAÇÃO

Declaro, sob compromisso de honra, que o trabalho apresentado neste projeto, com o título *"Desenvolvimento seguro de aplicação Web Smart Sign"*, é original e foi realizado por Estudante Fábio da Silva Baptista (2213268) sob orientação de Professor Doutor Sílvio Priem Mendes (smendes@ipleiria.pt), Professor Doutor Paulo Jorge Gonçalves Loureiro (paulo.loureiro@ipleiria.pt) e Professor Doutor Paulo Manuel Almeida Costa (paulo.costa@ipleiria.pt).

Leiria, 21 de Março de 2024

Assinado por: **Fábio da Silva Baptista**
Num. de Identificação: 15490654
Data: 2024.03.21 10:28:55+00'00'

Estudante Fábio da Silva Baptista